# Contents

## DIGITAL SYSTEMS (CS-BRANCH)

PAGE NO.

UNIT - I :

Review of number systems and number base conversions ....................(03 to 25)
Binary codes ....................(25 to 34)
Logic gates ....................(34 to 43)
Boolean algebra, Boolean functions, Simplification of Boolean functions ....................(43 to 56)
Karnaugh map methods, SOP-POS simplification, NAND-NOR implementation ....................(56 to 74)

UNIT - 2 :

Combinational Logic – Half adder, Half subtractor, Full adder, Full subtractor ....................(75 to 86)
Series and parallel addition, look-ahead carry generator, BCD adder .....(86 to 96)
Multiplexer-demultiplexer ....................(96 to 104)
Encoder-decoder, arithmetic circuits, ALU ....................(104 to 116)

UNIT - 3 :

Sequential logic – Flip-flops, D, T, S-R, J-K Master-Slave, racing condition, Edge & Level triggered circuits ....................(117 to 131)
Shift registers, Asynchronous and synchronous counters, their types and state diagrams ....................(131 to 170)
Semiconductor memories, Introduction to digital ICs 2716, 2732, etc. & their address decoding, Modern trends in semiconductor memories such as DRAM, FLASH RAM, etc., Designing with ROM and PLA .........(170 to 184)

UNIT - 4 :

Introduction to A/D & D/A converters & their types, sample and hold circuits, Voltage to Frequency & Frequency to Voltage conversion ......(185 to 208)
Multivibrators – Bistable, Monostable, Astable, Schmitt trigger, IC 555 & Its applications ....................(208 to 225)
TTL, PMOS, CMOS and NMOS logic, Interfacing between TTL to MOS..(225 to 248)

UNIT - 5 :

Introduction to Digital Communication, Nyquist sampling theorem, time division multiplexing ....................(249 to 261)
PCM, quantization error ....................(262 to 272)
Introduction to BPSK & BFSK modulation schemes, Shannon's theorem for channel capacity ....................(272 to 288)

# UNIT 1

## REVIEW OF NUMBER SYSTEMS AND NUMBER BASE CONVERSIONS

***Q.1. What do you understand by digital circuit and systems ?***
***(R.G.P.V., June 2009)***

***Ans.*** A ***digital circuit*** is one in which the voltage levels assume a finite number of distinct values.

Digital circuits are often called switching circuits, because the voltage levels in a digital circuit are assumed to be switched from one value to another instantaneously, that is the transition time is assumed to be zero.

Digital circuits are also called logic circuits, because each type of digital circuit obeys a certain set of logic rules. The manner in which a logic circuit responds to an input is referred to the circuit's logic.

All of us are familiar with the impact of modern digital computers, communication systems, digital display systems, internet, e-mail etc. on society. One of the main causes of this revolution is the advent of integrated circuits (ICs), which became possible because of the tremendous progress in semiconductor technology in recent years. Most of us may not be familiar with the principles of working of computers, communication systems, internet, e-mail, etc. even though these have become an important part of our daily life. The operation of these systems, and many other systems, is based on the principles of digital techniques and these systems are referred to as ***digital systems***.

***Q.2. Differentiate between analog and digital circuits.***
***(R.G.P.V., Nov. 2018)***

***Ans.*** Differences between analog and digital circuits are as follows –

| *S.No.* | ***Analog Circuits*** | ***Digital Circuits*** |
|---|---|---|
| (i) | Analog circuits are those in which voltages and currents vary continuously through the given range. | Digital circuits are those in which the voltage levels may consider only a finite number of distinct values. |

| | | |
|---|---|---|
| (ii) | These circuits operate on continuous valued signals. | These circuits operate on signals that exist only at two level, i.e. 0's and 1's. |
| (iii) | No conversion of input signals are required before processing, i.e. input signal is analog, the circuit directly performs various logical operations and produces an analog output. | In digital circuits, the input signals are converted from analog to digital form before it is processed, i.e. the digital circuit is capable of processing digital signals only, and produces output which is again converted back from digital to analog signals so that the output gives meaningfull results that can be understood by humans. |
| (iv) | In analog circuits, since there are no conversions involved at the input or at the output side there is no loss of information that is available for processing. | Due to the conversion process at the input side (analog to digital) and at the output side, some amount of information is lost during the conversion process. |
| (v) | The man power available to design analog circuits is very low, this results in long time to market the finished products. | The available man power to design digital circuits is significantly large compared to that of analog circuit designers. |
| (vi) | Analog circuits are mostly custom made and lacks flexibility. | Digital circuits have high degree of flexibility. |

***Q.3. What is meant by number systems ? Discuss its types.***

***Ans.*** Generally in any number system there is an ordered set of symbols called digits, which are used to specify any number. The digits are defined for performing arithmetic operations, such as addition, subtraction, multiplication, etc. A collection of these digits forms a number, which in general has two parts, namely integer and fractional. These two parts are set apart by a radix point (.).

In any number representation, the left most digit, which has the large positional weight out of all the digits shown in that number is known as the most significant bit (MSB) and the right most digit, which has least positional weight out of all the digits present in that number is known as the least significant bit (LSB).

Generally, four types of number systems are used in digital electronics as –

***(i) Binary Number System*** – It is a positional weighted system. The base of this number system is 2. Only two symbols, namely 0 and 1 are used in this system. These are called ***bits***. The binary number consists of a sequence of bits, each of which is either 0 or 1. In the binary number system, a group of four bits is called ***nibble*** and a group of 8-bits is called ***byte***.

***(ii) Octal Number System*** – The octal number system is also positional weighted system. The octal number system uses the digits 0, 1, 2, 3, 4, 5, 6 and 7. The base of this system is eight (8). Since its base 8 = $2^3$, every 3-bit group of binary can be represented by an octal digit. The least significant position has weight of $8^0$, i.e., 1, the higher significant positions are given weights in the ascending powers of eight (8), i.e., $8^1$, $8^2$, $8^3$, etc., respectively.

***(iii) Decimal Number System*** – It is a positional weighted system, which means that the value attached to a symbol depends on its location with respect to the decimal point.

The decimal number system contains ten unique symbols, 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9. Since counting in decimal involves ten symbols, its base is ten. The digits to the right of decimal point have weights, which are negative powers of 10 and forms fractional part. The digits to the left of the decimal point have weights, which are positive powers of 10 and forms integer part. The value of decimal number is the sum of the products of the digits of that number with their respective column weights.

Let us consider a mixed decimal number $(N)_b = d_n\ d_{n-1}\ d_{n-2}\ .....d_1 d_0\ d_{-1}\ d_{-2}\ d_{-3}\ .....d_{-k}$. The value of this mixed number is given by –

$$\begin{aligned}(N)_b = {} & (d_n \times 10^n) + (d_{n-1} \times 10^{n-1}) + .... + (d_1 \times 10^1) + (d_0 \times 10^0) \\ & + (d_{-1} \times 10^{-1}) + (d_{-2} \times 10^{-2}) + (d_{-3} \times 10^{-3}) + .... + (d_{-k} \times 10^{-k}).\end{aligned}$$

where $(N)_b$ denotes the value of entire number.

***(iv) Hexadecimal Number System*** – The base of hexadecimal number system is 16, i.e., it has 16 independent symbols. These 16 symbols are namely 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E and F. Since its base is 16 = $2^4$, every 4 binary digit combination can be represented by one hexadecimal digit. Each significant position in an hexadecimal number has a positional weight. The least significant position has a weight of $16^0$, i.e., 1, the higher significant positions are given weights in the ascending powers of 16, i.e., $16^1$, $16^2$, $16^3$, etc., respectively.

***Q.4. What do you mean by radix 7 ?***

***Ans.*** As we know that the decimal number system contains ten unique symbols 0, 1, 2 3, 4, 5, 6, 7, 8 and 9. Since counting in decimal involves ten symbols, we say that its base or radix is ten. Similarly the radix 7 means that it contains 7 symbols and its base or radix is seven.

The principle of positional weighting can be extended to any number system. Any number can be represented by the equation

$$Y = d_n r^n + d_{n-1} r^{n-1} + .... + d_1 r^1 + d_0 r^0$$

where Y is the value of the entire number, $d_n$ is the value of the $n^{th}$ digit from the point and r is the radix or base. This equation has already been applied to the different number system. Similarly we can apply it to the radix 7 systems.

*Q.5. Do the conversions with the help of example –*
*(i) Binary to octal (ii) Octal to binary.*

*Ans. (i) Binary to Octal* – The binary numbers can be converted into equivalent octal numbers by making groups of 3-bits starting from least significant bit and moving towards most significant bit for integer part of the number. For fractional part, the 3 bit grouping are made from the starting of binary (radix) point.

**Example** – $(110101.101010)_2 = (\ )_8$

Group of three bits are $\underbrace{110}_{6}\ \underbrace{101}_{5}\ .\ \underbrace{101}_{5}\ \underbrace{010}_{2}$

Convert each group to octal

The result is $\mathbf{(65.52)_8}$ **Ans.**

*(ii) Octal to Binary* – To convert an octal number to its equivalent binary number each digit of the given octal number is converted to its 3-bit binary equivalent.

**Example** – Convert the $(56.34)_8$ to its equivalent binary number.

$$(56.34)_8 = (101)\ (110).(011)\ (100)$$
$$= (101110.011100)_2$$

***Q.6. Explain the following conversions with the help of example –***

***(i) Binary to hexadecimal (ii) Hexadecimal to binary.***

***Ans. (i) Binary to Hexadecimal*** – Binary number can be converted into the equivalent hexadecimal numbers by making groups of four bits starting from LSB and moving towards MSB for integer part and then replacing each group of four bits by its hexadecimal representation. Consider an example to convert binary number $(1101\ 0010\ 110\ 111\ 0101)_2$ to hexadecimal number.

| | | | | |
|---|---|---|---|---|
| 0011 | 0100 | 1011 | 0111 | 0101 |
| 3 | 4 | B | 7 | 5 |

$= (34B75)_{16}$

***(ii) Hexadecimal to Binary*** – Hexadecimal numbers can be converted into equivalent binary numbers by replacing each next digit by its equivalent 4-bit binary number. This is represented by the example given below –

| | | | |
|---|---|---|---|
| $(A352.B1)_{16} =$ | 1010 | 0011 | 0101 |
| | A | 3 | 5 |
| | 0010 | 1011 | 0001 |
| | 2 | B | 1 |

Thus, $(A352.B1)_{16} = (1010001101010010.10110001)_2$

***Q.7. Explain the hexadecimal to decimal conversion and vice-versa with the help of example.***

***Ans.* Hexadecimal to Decimal** – The hexadecimal number to decimal number conversion is done by multiply each digit in the hexadecimal number with their weight of its position and add all the product terms.

**Example** – $(5E2C.7B)_{16} = (\ )_{10}$

$$(5E2C.7B)_{16} = 5 \times 16^3 + 14 \times 16^2 + 2 \times 16^1 + 12 \times 16^0 + 7 \times 16^{-1} + 11 \times 16^{-2}$$
$$= 20480 + 3584 + 32 + 12 + 0.4375 + 0.04297$$
$$= (24108.4805)_{10}$$

**Decimal to Hexadecimal** – For the conversion of a decimal number to an equivalent hexadecimal number, the decimal number is divided by 16 successively. For the conversion of decimal fraction to its equivalent hexadecimal fraction the technique of repeated multiplication by 16 is used. The integer part is note down after each multiplication and the new remainder fraction is used for multiplication at the next stage.

**Example** – Convert $(2586)_{10}$ to its hexadecimal equivalent.

| | **Quotient** | **Remainder** |
|---|---|---|
| 2586 ÷ 16 | 161 | 10 = A |
| 161 ÷ 16 | 10 | 1 |
| 10 ÷ 16 | 0 | 10 = A |

Thus $(2586)_{10} = (A1A)_{16}$

It may be noted that the remainder of the division processes from the digits of the hexadecimal number and the remainders that are greater than 9 are represented by the letters A through F.

***Q.8. Explain the following conversions with the help of example –***
***(i) Octal to hexadecimal (ii) Hexadecimal to octal.***

***Ans. (i) Octal to Hexadecimal*** – To convert an octal number to hexadecimal, the steps are as follows –

(a) Convert the given octal number to its binary equivalent

(b) Form groups of 4-bits, starting from the LSB

(c) Write the equivalent hexadecimal number for each group of 4-bits.

**Example** – Convert $(46.57)_8$ to its hexadecimal equivalent.

Converting $(46.57)_8$ first to its binary equivalent, we get

$$(46.57)_8 = (100)\ (110).\ (101)\ (111) = (100\ 110.\ 101111)_2$$

Now, forming the groups of 4 binary bits to obtain its hexadecimal equivalent we have

$$(100\ 110.101111) = (10)\ (0110).(1011)\ (11)$$
$$= (0010)\ (0110).(1011)\ (1100) = (26.BC)_{16}$$

***(ii) Hexadecimal to Octal*** – To convert a hexadecimal number to octal, the following steps can be applied –

(a) Convert the given hexadecimal number to its binary equivalent.

(b) Form groups of 3-bits, starting from the LSB.

(c) Write the equivalent octal number for each group of 3-bits.

**Example** – Convert $(47)_{16}$ to its octal equivalent.

$$(47)_{16} = (0100\ 0111)_2 = (01000111)_2 = (107)_8$$

Thus, 47 in hexadecimal is equivalent to 107 in the octal number system.

***Q.9. Explain the binary to decimal conversion and vice-versa.***

*Ans.* **Binary to Decimal** – To convert a binary number to its decimal equivalent we use the following expression –

The weight of the nth bit of the number from the right hand side = nth bit × $(\text{Base})^{n-1}$.

First we mark the bit position and then we give the weight of each bit of the number depending on its position. The sum of the weights of all bits gives the equivalent number.

**Example** – $(11101.110)_2 = (\ )_{10}$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Positional weights – | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ | $2^{-1}$ | $2^{-2}$ | $2^{-3}$ |
| Binary number – | 1 | 1 | 1 | 0 | 1 . | 1 | 1 | 0 |

$$(11101.110)_2 = (1 \times 2^4) + (1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0)$$
$$+ (1 \times 2^{-1}) + (1 \times 2^{-2}) + (0 \times 2^{-3})$$
$$= 16 + 8 + 4 + 0 + 1 + 0.5 + 0.25 + 0 = (29.75)_{10}$$

**Decimal to Binary** – There are two methods, which are used to convert a binary number to a decimal number, namely, sum of weights method and double dabble method.

In sum of weights method, the set of binary weight values whose sum is equal to the decimal number is determined.

In the double dabble method, the decimal integer number is converted to binary integer number by successive division of 2 and the decimal fraction is converted to binary fraction by successive multiplication of 2. In this method, the given decimal number is successively divided by 2 till the quotient is zero. The last remainder is the MSB. Thus, the integer numbers read from top to bottom give the equivalent binary fraction. To convert a mixed number to binary, convert the integer and fraction parts individually to binary and then combine them.

**Example** – Convert the decimal number 15.85 into binary.

| Integer Part | Quotient | Remainder |
|---|---|---|
| 15 ÷ 2 | 7 | 1 |
| 7 ÷ 2 | 3 | 1 |
| 3 ÷ 2 | 1 | 1 |
| 1 ÷ 2 | 0 | 1 |

↑ Read

15 (Decimal number) = 1111 (Binary number)

**Fractional Part –**

| *Fraction* | *Fraction × 2* | *Remainder New Fraction* | *Integer* |
|---|---|---|---|
| 0.85 | 1.7 | 0.7 | 1 (MSB) |
| 0.7 | 1.4 | 0.4 | 1 |
| 0.4 | 0.8 | 0.8 | 0 |
| 0.8 | 1.6 | 0.6 | 1 |
| 0.6 | 1.2 | 0.2 | 1 |
| 0.2 | 0.4 | 0.4 | 0 (LSB) |

Thus $(0.85)_{10} = (0.110110)_2$

Therefore, $(15.85)_{10} = (1111.110110)_2$

***Q.10. Explain the following conversions with the help of examples –***

***(i) Octal to decimal (ii) Decimal to octal.***

***Ans. (i) Octal Number into Decimal Number*** – The octal number to decimal number conversion is done by multiply each digit in the octal number with their weight of its position and add all the product terms.

**Example** – $(4154.24)_8 = (\ )_{10}$

$$(4154.24)_8 = (4 \times 8^3) + (1 \times 8^2) + (5 \times 8^1) + (4 \times 8^0)$$
$$+ (2 \times 8^{-1}) + (4 \times 8^{-2})$$
$$= (2048) + (64) + (40) + (4) + (0.25) + (0.0625)$$

Thus, $(4154.24)_8 = (2156.3125)_{10}$

***(ii) Decimal Number into Octal Number*** – To convert the given decimal integer number to octal number, successively divide the given number by factor 8 till the quotient is 0. The last remainder is the MSB. The remainder read from bottom to top give the equivalent octal integer number. To convert the given decimal fractional number to octal fractional number, successively multiply the decimal fractional number by factor 8 till the product is 0 or till the required accuracy is obtained. The first integer from the top is the MSB. The integer read downward to give the octal fractional number.

**Example** – Convert $(444.96)_{10}$

| **Integer Part** | **Quotient** | **Remainder** |
|---|---|---|
| 444 ÷ 8 | 55 | 4 |
| 55 ÷ 8 | 6 | 7 |
| 6 ÷ 8 | 0 | 6 |

Reading the remainders from bottom to top, the decimal number $(444)_{10}$ is equivalent to octal $(674)_8$

**Fractional Part –**

| *Fraction* | *Fraction × 8* | *Remainder New Fraction* | *Integer* |
|---|---|---|---|
| 0.96 | 7.68 | 0.68 | 7 (MSB) |
| 0.68 | 5.44 | 0.44 | 5 |
| 0.44 | 3.52 | 0.52 | 3 |
| 0.52 | 4.16 | 0.16 | 4 |
| 0.16 | 1.28 | 0.28 | 1 (LSB) |

This process will continue further, so may take the result upto 5 places of octal point

$$(0.96)_{10} = (0.75341)_8$$

Hence the result $\quad (444.96)_{10} = (674.75341)_8$

***Q.11. What do you mean by signed binary numbers ?***

***Ans.*** In case of signed binary number, positive integer including zero can be represented as unsigned number. To represent negative integer, we need a notation for negative values. In ordinary arithmetic a negative number is indicated by minus sign and a positive number by a plus sign. Because of computer hardware limitation, computers must represent every thing with binary digits. It is usually to represent the sign with a bit placed in the leftmost position of the number. The convention is to make the sign bit 0 for positive and 1 for negative.

For example, the string of bits 00101 can be considered as 5 (unsigned binary) or + 5 (signed binary) because the left most bit is 0. The string of bit 10101 represents the binary equivalent of 21 when considered as an unsigned number or as – 5 when considered as a signed number. This is because the 1 that is in the leftmost position designates a negative and the other 4-bits represent binary 5.

The signed magnitude, – 5 is obtained from + 5 by changing the sign bit in the leftmost position from 0 to 1. In signed – 1's complement, – 5 is obtained by complementing all the bits of + 5, including the sign bit. The signed – 2's complement representation of – 5 is obtained by taking the 2's complement of the positive number including the sign bit.

The signed – 2's complement system has only one representation for 0, which is always positive. The signed magnitude system is used in ordinary arithmetic, but is awkward if employed in computer arithmetic because of separate handling of the sign and the magnitude.

***Q.12. What do you mean by 1's complement representation ? Explain with example.***

***Ans.*** The 1's complement in the binary number system is similar to 9's complement in the decimal system. To obtain 1's complement of a binary number each bit of the binary number is subtracted from 1. Thus 1's complement of a binary number may be formed by simply changing each 1 to a 0 and each 0 to a 1.

**Example** – The 1's complement of the binary number 010 is 101.

***Q.13. What do you mean by 2's complement representation ? Explain with example.***

***Ans.*** The 2's complement in the binary number system is similar to 10's complement in the decimal number system. The 2's complement of a binary number is equal to the 1's complement of the number plus one.

The 2's complement of a binary number = Its 1's complement + 1.

**Example** – The 2's complement of 0101 = 1010 + 1 = 1011.

## NUMERICAL PROBLEMS

***Prob.1. Convert the following – (i) $(48.625)_{10} = ()_2$ (ii) Divide $(1EC87)_{16}$ by $(A5)_{16}$.*** ***(R.G.P.V., June 2014)***

***Sol.*** (i) Conversion of integer part $(48)_{10}$ –

| **Successive Division** | **Remainder** |
|---|---|
| 2 \| 48 | 0 (LSB) |
| 2 \| 24 | 0 |
| 2 \| 12 | 0 |
| 2 \| 6 | 0 |
| 2 \| 3 | 1 |
| 1 | 1 (MSB) |

i.e., $\quad (48)_{10} = (110000)_2$

Conversion of fractional part $(0.625)_{10}$ –

| | | |
|---|---|---|
| 0.625 × 2 | 0.25 × 2 | 0.50 × 2 |
| 1.25 | 0.50 | 1.00 |
| ↓ | ↓ | ↓ |
| 1 | 0 | 1 |

i.e., $(0.625)_{10} = (0.101)_2$

Hence, $\quad (48.625)_{10} = \mathbf{(110000.101)_2}$ **Ans.**

(ii) Divide $(1EC87)_{16}$ by $(A5)_{16}$

```
        2 F C
    A5)1 E C 8 7
       1 4 A
         A 2 8
         9 A B
           7 D 7
           7 B C
             1 B    Remainder
```

**Ans.**

***Prob.2. Convert the following –***

***(i)*** $(0.513)_{10}$ ***to octal*** ***(ii)*** $(673.124)_8$ ***to binary***

***(iii)*** $(1010.01101)_2$ ***to decimal.*** ***(R.G.P.V., Dec. 2010)***

***Sol.*** (i) $(0.513)_{10} = 0.513 \times 8 = 4.104$

$0.104 \times 8 = 0.832$

$0.832 \times 8 = 6.656$

$0.656 \times 8 = 5.248$

$0.248 \times 8 = 1.984$

$0.984 \times 8 = 7.872$

$(0.513)_{10} = \mathbf{(0.406517\ .....)_8}$ **Ans.**

(ii) $$(673.124)_8 = (\underset{\downarrow}{6}\ \ \underset{\downarrow}{7}\ \ \underset{\downarrow}{3}\ .\ \underset{\downarrow}{1}\ \ \underset{\downarrow}{2}\ \ \underset{\downarrow}{4})_8$$

$= (110\ 111\ 011\ .\ 001\ 010\ 100)_8$

$(673.124)_8 = \mathbf{(110111011.001010100)_2}$ **Ans.**

(iii) $(1010.01101)_2 = (1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0) + (0 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} + 0 \times 2^{-4} + 1 \times 2^{-5})$

$= 8 + 2 + 0.25 + 0.125 + 0.03125$

$= \mathbf{(10.40625)_{10}}$ **Ans.**

***Prob.3. Convert*** $(412)_{10}$ ***to –***

***(i) Binary (ii) Octal (iii) Hexadecimal.*** ***(R.G.P.V., Dec. 2017)***

***Sol.*** (i) $(412)_{10} = (?)_2$

| **Successive Division** | | **Remainder** |
|---|---|---|
| 2 | 412 | 0 (LSB) ↑ |
| 2 | 206 | 0 |
| 2 | 103 | 1 |
| 2 | 51 | 1 |
| 2 | 25 | 1 |
| 2 | 12 | 0 |
| 2 | 6 | 0 |
| 2 | 3 | 1 |
| 2 | 1 | 1 (MSB) |
| | 0 | |

Hence, $(412)_{10} = \mathbf{(110011100)_2}$ **Ans.**

(ii) $(412)_{10} = (?)_8$

| **Successive Division** | | **Remainder** |
|---|---|---|
| 8 | 412 | 4 (LSB) ↑ |
| 8 | 51 | 3 |
| 8 | 6 | 6 (MSB) |
| | 0 | |

Hence, $(412)_{10} = \mathbf{(634)_8}$ **Ans.**

(iii) $(412)_{10} = (?)_{16}$

| **Successive Division** | | **Remainder** |
|---|---|---|
| 16 | 412 | 12 = C |
| 16 | 25 | 9 = 9 |
| | 1 | 1 = 1 |

Hence, $(412)_{10} = \mathbf{(19C)_{16}}$ **Ans.**

***Prob.4. Convert the following –***

***(i) Decimal 225.225 to binary, octal and hexadecimal.***

***(ii) Binary 11010111.110 to decimal, octal and hexadecimal.***

***(R.G.P.V., June 2017)***

***Sol.*** (i) (a) $(225.225)_{10} = (?)_2$

Conversion of integer part $(225)_{10}$ –

| **Successive Division** | | **Remainder** |
|---|---|---|
| 2 | 225 | 1 (LSB) ↑ |
| 2 | 112 | 0 |
| 2 | 56 | 0 |
| 2 | 28 | 0 |
| 2 | 14 | 0 |
| 2 | 7 | 1 |
| 2 | 3 | 1 |
| | 1 | 1 (MSB) |

i.e., $(225)_{10} = (11100001)_2$

Conversion of fractional part $(0.225)_{10}$ –

$$\begin{array}{ccccc} \dfrac{0.225 \times 2}{0.45} & \rightarrow \dfrac{0.45 \times 2}{0.90} & \rightarrow \dfrac{0.90 \times 2}{1.80} & \rightarrow \dfrac{0.80 \times 2}{1.60} & \rightarrow \dfrac{0.60 \times 2}{1.20} \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ 0 & 0 & 1 & 1 & 1 \end{array}$$

i.e., $(0.225)_{10} = (0.0011)_2$

Hence, $(225.225)_{10} = \mathbf{(11100001.0011.....)_2}$ **Ans.**

(b) $(225.225)_{10} = (?)_8$

Conversion of integer part $(225)_{10}$ –

**Successive Division** **Remainder**

$$\begin{array}{r|l c c}
8 & 225 & & \\ \hline
8 & 28 & \rightarrow & 1 \uparrow \\ \hline
8 & 3 & \rightarrow & 4 \\ \hline
 & 0 & \rightarrow & 3
\end{array}$$

i.e., $(225)_{10} = (341)_8$.

Conversion of fractional part $(0.225)_{10}$ –

| Multiplication | Generated Integer |
|---|---|
| $0.225 \times 8 = 1.8$ | 1 ↓ |
| $0.8 \times 8 = 6.4$ | 6 |
| $0.4 \times 8 = 3.2$ | 3 |
| $0.2 \times 8 = 1.6$ | 1 |
| $0.6 \times 8 = 4.8$ | 4 |

Hence, $(225.225)_{10}$ = $\mathbf{(341.16314.....)_8}$ **Ans.**

(c) $(225.225)_{10} = (?)_{16}$

Conversion of integer part $(225)_{10}$ –

**Successive Division** **Remainder**

$$\begin{array}{r|l c}
16 & 225 & 1 = 1 \uparrow \\ \hline
16 & 14 & 14 = E \\ \hline
8 & 0 &
\end{array}$$

i.e., $(225)_{10} = (E1)_{16}$

Conversion of fractional part $(0.225)_{10}$ –

| | |
|---|---|
| $0.225 \times 16$ | 3.6 ↓ |
| $0.600 \times 16$ | 9.6 |
| $0.600 \times 16$ | 9.6 |

i.e. $(0.225)_{10} = (0.399.....)_{16}$

Hence,

$(225.225)_{10}$ = $\mathbf{(E1.399.....)_{16}}$ **Ans.**

(ii) (a) $(11010111.110)_2 = (\ )_{10}$

$$= 1 \times 2^7 + 1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3}$$

$$= 128 + 64 + 0 + 16 + 0 + 4 + 2 + 1 + 0.5 + 0.25 + 0$$

$= \mathbf{(215.75)_{10}}$ **Ans.**

(b) $(11010111.110)_2 = (\ )_8$ – To convert given binary number into octal number, grouping this binary number into 3-bit equivalent octal digit. Therefore,

$$= \left(\underbrace{011}_{3}\ \underbrace{010}_{2}\ \underbrace{111}_{7} \cdot \underbrace{110}_{6}\right)_2 = \mathbf{(327.6)_8}$$ **Ans.**

(c) $(11010111.110)_2 = (\ )_{16}$

$$= \left(\underbrace{1101}_{D}\ \underbrace{0111}_{7} \cdot \underbrace{1100}_{C}\right)_2$$

Hence, $(11010111.10)_2 = \mathbf{(D7.C)_{16}}$ **Ans.**

***Prob.5. Convert the number $(210.25)_{10}$ to base 2, 8.*** ***(R.G.P.V., June 2015)***

***Sol.*** (i) Given, $(210.25)_{10} = (\ )_2$

Conversion of integer part $(210)_{10}$ –

**Successive Division** **Remainder**

$$\begin{array}{r|l c}
2 & 210 & 0 \ (LSB) \\ \hline
2 & 105 & 1 \uparrow \\ \hline
2 & 52 & 0 \\ \hline
2 & 26 & 0 \\ \hline
2 & 13 & 1 \\ \hline
2 & 6 & 0 \\ \hline
2 & 3 & 1 \\ \hline
2 & 1 & 1 \ (MSB) \\ \hline
 & 0 &
\end{array}$$

i.e., $(210)_{10} = (11010010)_2$

Conversion of fractional part $(0.25)_{10}$ –

$$\begin{array}{c c c}
0.25 \times 2 & & 0.50 \times 2 \\ \hline
0.50 & \longrightarrow & 1.00 \\
\downarrow & & \downarrow \\
0 & & 1
\end{array}$$

i.e., $(0.25)_{10} = (0.01)_2$

Hence, $(210.25)_{10}$ = $\mathbf{(11010010.01)_2}$ **Ans.**

(ii) $(210.25)_{10} = (\ )_8$

Conversion of integer part $(210)_{10}$ –

**Successive Division** **Remainder**

$$\begin{array}{r|l c}
8 & 210 & 2 \ (LSB) \\ \hline
8 & 26 & 2 \uparrow \\ \hline
8 & 3 & 3 \ (MSB) \\ \hline
 & 0 &
\end{array}$$

i.e., $(210)_{10} = (322)_8$

Conversion of fractional part $(0.25)_{10}$ –

$$\begin{array}{c} 0.25 \times 8 \\ \hline 2.00 \\ \downarrow \\ 2 \end{array}$$

i.e., $(0.25)_{10} = (0.2)_8$

Hence, $(210.25)_{10} = \mathbf{(322.2)_8}$ **Ans.**

***Prob.6. Convert $(41.6875)_{10}$ to***

***(i) Binary (ii) Octal (iii) Hexadecimal.***

***(R.G.P.V., May 2018)***

***Sol.*** (i) Given, $(41.6875)_{10} = (\ )_2$

Conversion of integer part $(41)_{10}$ –

| Successive Division | | Remainder |
|---|---|---|
| 2 | 41 | 1 (LSB) |
| 2 | 20 | 0 |
| 2 | 10 | 0 |
| 2 | 5 | 1 |
| 2 | 2 | 0 |
| 2 | 1 | 1 (MSB) |

i.e. $(41)_{10} = (101001)_2$

Conversion of fractional part $(0.6875)_{10}$ –

$$\begin{array}{c} 0.6875 \times 2 \\ \hline 1.375 \\ \downarrow \\ 1 \end{array} \rightarrow \begin{array}{c} 0.375 \times 2 \\ \hline 0.75 \\ \downarrow \\ 0 \end{array} \rightarrow \begin{array}{c} 0.75 \times 2 \\ \hline 1.50 \\ \downarrow \\ 1 \end{array} \rightarrow \begin{array}{c} 0.50 \times 2 \\ \hline 1.0 \\ \downarrow \\ 1 \end{array}$$

i.e., $(0.6875)_{10} = (0.1011)_2$

Hence $(41.6875)_{10} = \mathbf{(101001.1011)_2}$ **Ans.**

(ii) $(41.6875)_{10} = (\ )_8$

Conversion of integer part $(41)_{10}$ –

| Successive Division | | Remainder |
|---|---|---|
| 8 | 41 | 1 |
| 8 | 5 | 5 |
| | 0 | |

i.e., $(41)_{10} = (51)_8$

Conversion of fractional part $(0.6875)_{10}$

$$\begin{array}{c} 0.6875 \times 8 \\ \hline 5.5 \\ \downarrow \\ 5 \end{array} \rightarrow \begin{array}{c} 0.5 \times 8 \\ \hline 4.0 \\ \downarrow \\ 4 \end{array}$$

i.e., $(0.6875)_{10} = (0.54)_8$

Hence, $(41.6875)_{10} = \mathbf{(51.54)_8}$ **Ans.**

(iii) $(41.6875)_{10} = (\ )_{16}$

Conversion of integer part $(41)_{10}$

| Successive Division | | Remainder |
|---|---|---|
| 16 | 41 | 9 |
| 16 | 2 | 2 |
| | 0 | |

i.e., $(41)_{10} = (29)_{16}$

Conversion of fractional part $(0.6875)_{10}$

$$\begin{array}{c} 0.6875 \times 16 \\ \hline 11.00 \\ \downarrow \\ B \end{array}$$

Hence, $(41.6875)_{10} = \mathbf{(29.B)_{16}}$ **Ans.**

***Prob.7. Convert the following –***

***(i) $(B65F)_{16} = (\ )_{10}$ (ii) $(153.25)_{10} = (\ )_2$***

***(iii) $(10110001101011)_2 = (\ )_8$ (iv) $(153)_{10} = (\ )_8$.***

***(R.G.P.V., June 2010)***

***Sol.*** (i)

$$\begin{aligned}(B65F)_{16} &= B \times 16^3 + 6 \times 16^2 + 5 \times 16^1 + F \times 16^0 \\ &= 11 \times 16^3 + 6 \times 16^2 + 5 \times 16 + 15 \times 1 \\ &= \mathbf{(46687)_{10}}\end{aligned}$$

**Ans.**

(ii) $(153.25)_{10} = (\ )_2$

| Successive Division | | Remainder |
|---|---|---|
| 2 | 153 | 1 LSB |
| 2 | 76 | 0 |
| 2 | 38 | 0 |
| 2 | 19 | 1 |
| 2 | 9 | 1 |
| 2 | 4 | 0 |
| 2 | 2 | 0 |
| 2 | 1 | 1 MSB |
| | 0 | |

$(153)_{10} = (10011001)_2$

Conversion of fractional part $(.25)_{10}$ –

$$\begin{array}{c} .25 \times 2 \\ \hline 0.50 \\ \downarrow \\ 0 \end{array} \rightarrow \begin{array}{c} .50 \times 2 \\ \hline 1.00 \\ \downarrow \\ 1 \end{array}$$

$(.25)_{10} = (.01)_2$

Thus $(153.25)_{10} = \mathbf{(10011001.01)_2}$ Ans

(iii) $(10110001101011)_2 = \underbrace{010}_{2}\ \underbrace{110}_{6}\ \underbrace{001}_{1}\ \underbrace{101}_{5}\ \underbrace{011}_{3}$

$= \mathbf{(26153)_8}$ Ans

(iv) $(153)_{10} = (\ )_8$

| | | Remainder |
|---|---|---|
| 8 | 153 | 1 ↑ |
| 8 | 19 | 3 |
| 8 | 2 | 2 |
| | 0 | |

$(153)_{10} = \mathbf{(231)_8}$ Ans

***Prob.8. Convert –***

***(i) 0.1011 → Decimal (Binary)*** ***(ii) $23_{(8)}$ → Decimal***

***(iii) $(9AF)_{(16)}$ → Binary.*** ***(R.G.P.V., Dec. 201***

***Sol.*** (i) $(0.1011)_2 = 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} + 1 \times 2^{-4}$

$= 0.5 + 0.125 + 0.0625 = \mathbf{(0.6875)_{10}}$ Ans

(ii) $23_{(8)} = 2 \times 8^1 + 3 \times 8^0 = 16 + 3 = \mathbf{(19)_{10}}$ Ans

(iii) $(9AF)_{16} = (\underset{\downarrow}{9}\quad \underset{\downarrow}{A}\quad \underset{\downarrow}{F})$

$(1001\quad 1010\quad 1111)_2$

$= \mathbf{(100110101111)_2}$ Ans

***Prob.9. Convert the following –***

***(i) $(4213.321)_5 = (\ )_{10}$*** ***(ii) $(10101101.110101001) = (\ )_8$***

***(iii) $(715230.3214)_8 = (\ )_{16}$*** ***(iv) $(7524.4251)_{10} = (\ )_2$.***

***(R.G.P.V., June 201***

***Sol.*** (i) $(4213.321)_5 = 4 \times 5^3 + 2 \times 5^2 + 1 \times 5^1 + 3 \times 5^0 + 3 \times 5^{-1} +$
$\times 5^{-2} + 1 \times 5$

$= 500 + 50 + 5 + 3 + \frac{3}{5} + \frac{2}{25} + \frac{1}{125}$

$= \mathbf{(558.688)_{10}}$ Ans

(ii) $(10101101.110101001)_2 = (\underbrace{010}_{2}\underbrace{101}_{5}\underbrace{101}_{5}.\underbrace{110}_{6}\underbrace{101}_{5}\underbrace{001}_{1})$

$= \mathbf{(255.651)_8}$ Ans

(iii) $(715230.3214)_8 = 111\ 001\ 101\ 010\ 011\ 000\ .\ 011\ 010\ 001\ 1$

$= \underbrace{0011}_{3}\underbrace{1001}_{9}\underbrace{1010}_{A}\underbrace{1001}_{9}\underbrace{1000}_{8}.\underbrace{0110}_{6}\underbrace{1000}_{8}\underbrace{1100}_{C}$

$= \mathbf{(39A98.68C)_{16}}$ Ans

(iv) $(7524.4251)_{10} = (\ )_2$

Conversion of integer $(7524)_{10}$ –

| | Successive Division | Remainder |
|---|---|---|
| 2 | 7524 | 0 (LSB) |
| 2 | 3762 | 0 |
| 2 | 1881 | 1 |
| 2 | 940 | 0 |
| 2 | 470 | 0 |
| 2 | 235 | 1 |
| 2 | 117 | 1 |
| 2 | 58 | 0 |
| 2 | 29 | 1 |
| 2 | 14 | 0 |
| 2 | 7 | 1 |
| 2 | 3 | 1 |
| 2 | 1 | 1 (MSB) |
| | 0 | |

i.e., $(7524)_{10} = (1110101100100)_2$

Conversion of fraction $(0.4251)_{10}$ –

$$\begin{array}{ccccc} 0.4251 \times 2 & 0.8502 \times 2 & 0.7004 \times 2 & 0.4008 \times 2 & 0.8016 \times 2 \\ \hline 0.8502 & 1.7004 & 1.4008 & 0.8016 & 1.6032 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ 0 & 1 & 1 & 0 & 1 \end{array}$$

i.e., $(0.4251)_{10} = (0.01101\ .....\ )_2$

Therefore the final result is

$(7524.4251)_{10} = \mathbf{(1110101100100.011011\ .....\ )_2}$ **Ans.**

***Prob.10. Convert the following –***

***(i) $(1111)_2 = (\quad)_{10}$*** ***(ii) $(10010.1011)_2 = (\quad)_{10}$***

***(iii) $(23)_{10} = (\quad)_2$*** ***(iv) $(5.5)_{10} = (\quad)_2$***

***(v) $(47.6)_{10} = (\quad)_2$***

***(R.G.P.V., Nov. 2018)***

***Sol.*** (i) $(1111)_2 = (\quad)_{10}$

$(1111)_2 = 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

$= 8 + 4 + 2 + 1$

$= \mathbf{(15)_{10}}$ **Ans.**

(ii) $(10010.1011)_2 = 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 +$
$1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} + 1 \times 2^{-4}$

$= 16 + 0 + 0 + 2 + 0 + 0.5 + 0 + 0.125 + 0.0625$

$= \mathbf{(18.6875)_{10}}$ **Ans.**

(iii) $(23)_{10} = (\quad)_2$

**Successive Division** **Remainder**

$$\begin{array}{r|l} 2 & 23 \\ \hline 2 & 11 \\ \hline 2 & 5 \\ \hline 2 & 2 \\ \hline 2 & 1 \\ \hline & 0 \end{array} \qquad \begin{array}{l} 1 \text{ (LSB)} \\ 1 \\ 1 \\ 0 \\ 1 \text{ (MSB)} \end{array} \uparrow$$

Hence $(23)_{10} = \mathbf{(10111)_2}$

(iv) $(5.5)_{10} = (\quad)_2$

Conversion of integer part $(5)_{10}$ –

**Successive Division** **Remainder**

$$\begin{array}{r|l} 2 & 5 \\ \hline 2 & 2 \\ \hline 2 & 1 \\ \hline & 0 \end{array} \qquad \begin{array}{l} 1 \\ 0 \\ 1 \end{array} \uparrow$$

i.e., $(5)_{10} = (101)_2$

Conversion of fractional part $(0.5)_{10}$ –

$$\frac{0.5 \times 2}{1}$$

Hence, $(5.5)_{10} = \mathbf{(101.1)_2}$

(v) $(47.6)_{10} = (\quad)_2$

Conversion of integer part $(47)_{10}$ –

**Successive Division** **Remainder**

$$\begin{array}{r|l} 2 & 47 \\ \hline 2 & 23 \\ \hline 2 & 11 \\ \hline 2 & 5 \\ \hline 2 & 2 \\ \hline 2 & 1 \\ \hline & 0 \end{array} \qquad \begin{array}{l} 1 \text{ LSB} \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \text{ MSB} \end{array} \uparrow$$

i.e., $(47)_{10} = (101111)_2$

Conversion of fractional part $(0.6)_{10}$

$$\frac{0.6 \times 2}{1.2} \to \frac{0.2 \times 2}{0.4} \to \frac{0.4 \times 2}{0.8} \to \frac{0.8 \times 2}{1.6}$$

$$\downarrow 1 \qquad \downarrow 0 \qquad \downarrow 0 \qquad \downarrow 1$$

Hence $(47.6)_{10} = \mathbf{(101111.1001)_{10}}$

***Prob.11. Do as directed –***

***(i) $(56)_{16} = (?)_{10}$***

***(ii) $(32)_{10} = (?)_2$***

***(iii) Bubbled OR gate is also called ............ .***

***(R.G.P.V., Dec. 2016)***

***Sol.*** (i) $(56)_{16} = 5 \times 16^1 + 6 \times 16^0$

$= \mathbf{(86)_{10}}$ **Ans.**

(ii) $(32)_{10} = (\ )_2$

**Successive Division** **Remainder**

$$\begin{array}{r|l} 2 & 32 \\ \hline 2 & 16 \\ \hline 2 & 8 \\ \hline 2 & 4 \\ \hline 2 & 2 \\ \hline 2 & 1 \\ \hline & 0 \end{array} \qquad \begin{array}{l} 0 \text{ LSB} \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \text{ MSB} \end{array} \uparrow$$

Hence $(32)_{10} = \mathbf{(100000)_2}$ **Ans.**

(iii) Bubbled OR gate is also called **NOR gate.** **Ans.**

***Prob.12. Multiply $(1AB)_{16}$ by $(89)_{16}$.*** ***(R.G.P.V., Feb. 2010)***

***Sol.*** First change given hexadecimal number into binary then multiply it

$(1\ AB)_{16} = (0001\ 1010\ 1011)_2$

$(89)_{16} = (1000\ 1001)_2$

```
000110101011 × 10001001
-----------------------
       000110101011
      000000000000×
     000000000000×
    000110101011×
   000000000000×
  000000000000×
 000000000000×
000110101011×
-------------------
0001110010010000011
-------------------
```

$(1110010010000011)_2$

Now change this binary equivalent into hexadecimal.

$$\underbrace{1110}_{\downarrow\, E}\ \underbrace{0100}_{\downarrow\, 4}\ \underbrace{1000}_{\downarrow\, 8}\ \underbrace{0011}_{\downarrow\, 3}$$

Hence,

$(1AB)_{16} \times (89)_{16} = \mathbf{(E483)_{16}}$ **Ans.**

***Prob.13. Subtract using 1's and 2's complement –***
***(i) 10110101 – 01101101 (ii) 0010101 – 1101001.***
***(R.G.P.V., June 2***

***Sol. (i) Subtraction using 1's complement –***

```
          1 0 1 1 0 1 0 1
        + 1 0 0 1 0 0 1 0    (1's complement form)
Carry (1) 0 1 0 0 0 1 1 1
Add carry               + 1
          0 1 0 0 1 0 0 0
```

**Subtraction using 2's Complement –**

```
Minuend = 1 0 1 1 0 1 0 1
Subtrahend = 0 1 1 0 1 1 0 1
                  ↓
          1 0 0 1 0 0 1 0 (1's complement form)
                      + 1
          1 0 0 1 0 0 1 1 (2's complement form)
      1 0 1 1 0 1 0 1
    + 1 0 0 1 0 0 1 1  (2's complement form)
Carry (1) 0 1 0 0 1 0 0 0
```

The carry is discarded. The MSB is 0, so the result is positive and i true binary form

= **0 1 0 0 1 0 0 0**

***(ii) Subtraction using 1's complement –***

```
  0 0 1 0 1 0 1
+ 0 0 1 0 1 1 0 (1's complement form)
  0 1 0 1 0 1 1
```

No carry is obtained. The answer is 1's complement of 0101011 and opposite in sign i.e. – **1010100**

**Subtraction using 2's complement –**

```
Minuend = 0 0 1 0 1 0 1
Subtrahend = 1 1 0 1 0 0 1
                  ↓
          0 0 1 0 1 1 0 (1's complement form
                      + 1
          0 0 1 0 1 1 1 (2's complement form
  0 0 1 0 1 0 1
+ 0 0 1 0 1 1 1 (2's complement form)
  0 1 0 1 1 0 0
```

The MSB is 0, so the result is positive and is in true binary form

= **0 1 0 1 1 0 0**

***Prob.14. Subtract (1010)$_2$ from (1000)$_2$ using 1's and 2's complement method.*** ***(R.G.P.V., Dec. 2014)***

***Sol.*** Using 1's complement method –

```
                    1 0 0 0
1's Complement →  0 1 0 1
                    1 1 0 1
```

No carry is obtained. Then the answer is 1's complement of 1101 and is opposite in sign i.e. – **0010**. **Ans.**

Using 2's complement method –

```
                    1 0 0 0
2's Complement →  0 1 1 0
                    1 1 1 0
```

No carry is obtained. Thus, the difference is negative and the true answer is the 2's complement of (1110)$_2$ i.e. **(0010)$_2$**. **Ans.**

***Prob.15. Subtract (595)$_{10}$ from (378)$_{10}$.*** ***(R.G.P.V., June 2009)***

***Sol.*** The binary equivalent of the given decimal numbers are –

$$(378)_{10} = (000101111010)_2$$

$$(595)_{10} = (001001010011)_2$$

The subtraction of two given decimal numbers are given below –

```
  (378)10        000101111010 → Minuend
– (595)10  ⇒  + 110110101101 → 2's complement of subtrahend
   –217          111100100111
```

Here the final carry is 0, the result is negative (the minuend is smaller than the subtrahend) and is in 2's complement form. The 2's complement of 111100100111 = 000011011001, which is equal to 217. Therefore, the result is **(– 217)$_{10}$** **Ans.**

***Prob.16. Add and subtract octal numbers 360 and 715.***
***(R.G.P.V., Dec. 2015)***

***Sol. (i) Add –***

```
    3 6 0      =   0 1 1 1 1 0 0 0 0
+   7 1 5      =   1 1 1 0 0 1 1 0 1
 (1 2 7 5)8        1 0 1 0 1 1 1 1 0 1
```

***(ii) Subtract –***

```
  (3 6 0)8   =   0 1 1 1 1 0 0 0 0
– (7 1 5)8   =   0 0 0 1 1 0 0 1 1 ← 2's complement of (715)8
– (3 3 5)8       1 0 0 1 0 0 0 1 1
```

2's complement of 100100011 = 011011101 = (335)$_8$

***Prob.17. Using 9's complement, subtract (63458 – 3354).*** ***(R.G.P.V., Dec. 20***

***Sol.*** Regular subtraction

$$\begin{array}{r} 63458\\ -\ \ 3354\\ \hline 60104 \end{array}$$

9's complement subtraction

$$\begin{array}{rl} 63458 & \\ +\ \ 6645 & \text{9's complement of 3354}\\ \hline 70103 & \\ \downarrow & \\ -29896 & \text{9's complement of 70103} \end{array}$$

Therefore the answer is – **29896** A

***Prob.18. Subtract 49 from 34 using 9's complement. Also perform di subtraction for comparison.*** ***(R.G.P.V., Nov. 20***

***Sol.***

Regular Subtraction

$$\begin{array}{r} 34\\ -49\\ \hline -15 \end{array}$$

9's Complement Subtraction

$$\begin{array}{rl} 34 & \\ +50 & \longleftarrow \text{9's complement of 49 (i.e. 99 – 49 =}\\ \hline 84 & \\ \downarrow & \\ -15 & \longleftarrow \text{9's complement of 84} \end{array}$$

Therefore the answer is – **15.** A

***Prob.19. Subtract the following number using 10's complement metho***

***786 – 427.***

***(R.G.P.V., Nov./Dec. 20***

***Sol.*** Given, Minuend = 786

Subtrahend = 427

$$\text{9's complement of subtrahend } = \begin{array}{r} 9\ 9\ 9\\ -\ 4\ 2\ 7\\ \hline 5\ 7\ 2 \end{array}$$

$$\text{10's complement of subtrahend } = \begin{array}{r} 5\ 7\ 2\\ +\qquad 1\\ \hline 5\ 7\ 3 \end{array}$$

Add this to minuend –

$$\begin{array}{r} 7\ 8\ 6\\ +\ 5\ 7\ 3\\ \hline \text{End-around Carry (EAC)} \longleftarrow \boxed{1}\ 3\ 5\ 9 \end{array}$$

$\uparrow$ Ignore

Therefore the result is **359** A

***Prob.20. Subtract the following –***

***(i) FB2 – DAB (ii) 53BA – 2BCD (iii) 113 – 57***

***(R.G.P.V., Nov. 20***

***Sol.*** (i) The hex. subtraction is performed as explained below –

$$\begin{array}{ccc} 16^2 & 16^1 & 16^0\\ F & B & 2\\ -\ D & A & B\\ \hline 2 & 0 & 7\\ \hline \end{array}$$

In the $16^0$'s column, B cannot be subtracted from 2. So, borrow a 1 from the $16^1$'s column. The 1 borrowed from the $16^1$'s column becomes 10 in that column.

In the $16^0$'s column, $16 + 2 - B = 18 - 11 = 7$

In the $16^1$'s column, $10 - A = 10 - 10 = 0$

In the $16^2$'s column, $F - D = 15 - 13 = 2$

The result is $\mathbf{(207)_{16}}$. **Ans.**

(ii)

$$\begin{array}{cccc} 5 & 3 & B & A\\ -\ 2 & B & C & D\\ \hline \mathbf{2} & \mathbf{7} & \mathbf{E} & \mathbf{D}\\ \hline \end{array}$$

**Ans.**

(iii)

$$\begin{array}{rl} 113 & = 01001011\\ -\ \ 57 & = \underline{11010001} \longleftarrow \text{2's complement of } (57)_8 \text{ or } (00101111)\\ \hline (34)_8 & \quad 100011100\\ & \quad \uparrow\\ & \quad \text{carry discarded} \end{array}$$

Theresult is $\mathbf{(34)_8}$ **Ans.**

## BINARY CODES

***Q.14. Explain number systems and codes.*** ***(R.G.P.V., June 2011)***

***Ans.*** **Number Systems** – Refer the ans. of Q.3.

**Codes** – Code is a symbolic representation of discrete information, which may be present in the form of numbers, letters or physical quantities. The symbols used are the binary digits 0 and 1 which are arranged according to the rules of codes. These codes are used to communicate information to a digital computer and to retrieve messages from it. A code is used to enable an operator to feed data into a computer directly, in the form of decimal numbers, alphabets and special characters. The computer converts these data into binary codes and after, computation, transforms the data into its original format. When numbers, letters or words are represented by a special group of symbols, this is called encoding and the group of symbols is called a ***code.***

Some important groups of codes are given below –

(i) Weighted binary codes (ii) Non-weighted codes
(iii) Error-detecting codes (iv) Error-correcting codes
(v) Alphanumeric codes.

***Q.15. Explain the difference between a weighted and non-weig code with example.*** ***(R.G.P.V., June 2***

**Table 1.1 Some Weighted 4-bit Binary C**

| *Decimal Numbers* | *BCD or 8421 Code* | *2421 Code* | *542<br>Co* |
|---|---|---|---|
| 0 | 0000 | 0000 | 000 |
| 1 | 0001 | 0001 | 00 |
| 2 | 0010 | 0010 | 001 |
| 3 | 0011 | 0011 | 001 |
| 4 | 0100 | 0100 | 010 |
| 5 | 0101 | 1011 | 100 |
| 6 | 0110 | 1100 | 100 |
| 7 | 0111 | 1101 | 101 |
| 8 | 1000 | 1110 | 101 |
| 9 | 1001 | 1111 | 110 |

*Ans.* **Weighted Codes** – The codes which follow the positional weighting principles are known as weighted binary codes. Each position of a number represents a specific weight. In a weighted binary code the bits are multiplied by the weights indicated, the sum of these weighted bits give the equivalent decimal digit. Some weighted 4-bit binary codes with their decimal numbers is shown in table 1.1.

*(i)* ***BCD Code*** – The binary coded decimal uses the binary nu system to specify the decimal numbers 0 to 9. The weights are assi according to the positions occupied by these digits. The weights of the r most (or first) position is $2^0$ (or 1), the second position is $2^1$ (or 2), the t position is $2^2$ (or 4) and the fourth position is $2^3$ (or 8). The weights are 2-1 as reading from left to right and hence it is known as 8421 code.

*(ii)* ***2421 Code*** – The 2421 code is weighted binary code and its weig are 2,4,2 and 1. A decimal number is represented in 4-bit form and the total we of the four-bit = 2 + 4 + 2 + 1 = 9.

**Table 1.2 Gray Code Illustration f Decimal Numbers with Binary Co**

| *Decimal Numbers* | *Binary Code* $B_3B_2B_1B_0$ | *Gray Cod* $G_3G_2G_1$ |
|---|---|---|
| 0 | 0000 | 0000 |
| 1 | 0001 | 0001 |
| 2 | 0010 | 0011 |
| 3 | 0011 | 0010 |
| 4 | 0100 | 0110 |
| 5 | 0101 | 0111 |
| 6 | 0110 | 0101 |
| 7 | 0111 | 0100 |
| 8 | 1000 | 1100 |
| 9 | 1001 | 1101 |
| 10 | 1010 | 1111 |
| 11 | 1011 | 1110 |
| 12 | 1100 | 1010 |
| 13 | 1101 | 1011 |
| 14 | 1110 | 1001 |
| 15 | 1111 | 1000 |

Thus, the 2421 code shows decimal number from 0 to 9. This code is also a self complementing code, i.e., the 9's complement of number 'n' is obtained by complementing the 1's and 0's in the code word.

**Non-weighted Codes** – The codes which are not positionally weighted is called non-weighted codes. Means of that each position within a binary number is not assigned a fixed values. The Gray codes and Excess-3 codes are non-weighted codes.

*(i)* ***Gray Code*** – The Gray code is very useful code in which a decimal number is represented in binary form in such a manner, so each Gray code

number differs from the preceding and the succeeding number by a one-bit. The Gray code is a non-weighted code and therefore it is not suitable for arithmetic operations. The Gray code is applicable in input/output devices and in some types of analog to digital converters. The Gray code is a reflective digital code which contains a special property of containing two adjacent code numbers that differ by only one-bit. The Gray code is also known as ***unit distance code***. The Gray code is used where the normal sequence of binary numbers may produce an error during the transition from one number to the next number.

The Gray code illustration for the decimal number together with binary code is depicted in table 1.2.

*(ii)* ***Excess-3 Code*** – This code itself is complementing code which means that the 1's complement of the coded number gives 9's complement of the number itself. An Excess-3 code is a reflective code and is obtained by adding 3 to a decimal number. For example, Excess-3 code of decimal 2 is 0101, its 1's complement is 1010 which is Excess-3 for decimal 7. This is 9's complement of decimal 2. The self complementary property of Excess-3 code helps in performing subtraction operation in digital systems. Table 1.3 illustrates the BCD code, Excess-3 code for decimal numbers.

**Table 1.3 Illustration of Excess-3 and BCD Codes for Decimal Numbers**

| *Decimal* | *BCD Code* | | | | *Excess-3 Code* | | | |
|---|---|---|---|---|---|---|---|---|
| *Numbers* | *D* | *C* | *B* | *A* | $E_3$ | $E_2$ | $E_1$ | $E_0$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 2 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 3 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 4 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 5 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 6 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 7 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| 8 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| 9 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |

***Q.16. Justify the following statements with examples –***

***(i) Excess-3 code is self complementary code.***

***(ii) Gray code is a reflected code.***

***(R.G.P.V., May 2018)***

*Ans.* Refer the ans. of Q.15.

***Q.17. What do you mean by radix 7 ? Also explain reflected code.*** ***(R.G.P.V., June 2009)***

*Ans.* **Radix 7** – Refer the ans. of Q.4.

**Reflected Code** – When the code for 9 is the complement of the for 0, 8 for 1, 7 for 2, 6 for 3 and 5 for 4 then a code is said to be refle

***Q.18. Explain the self complementing code with example.*** ***(R.G.P.V., Dec. 2***

***Ans.*** A code is said to be self complementing, if the code word o complement of N, i.e., of 9-N can be obtained from code word of interchanging all the 0's and 1's. Therefore in self complementing code code for 9 is the complement of the code for 0, 8 for 1, 7 for 2, 6 for 3, 5 for 4. The 2421, 5211, 642-3 and XS-3 are self complementing codes

For a code to be self complementing sum of all its weight must be 9. is because whatever be the weight, 0 is to be represented by 0000 and sin a self complementing code, the code for 9 is the complement of the code 0, 9 has to be represented by 1111. There are only four (2421, 5211, 3 4311) positively-weighted, self complementing codes. There are 13 negati weighted self complementing codes.

***Q.19. What are the alphanumeric codes ?*** ***(R.G.P.V., June 2***

***Ans.*** An alphanumeric (sometimes abbreviated alphameric) code is a bi code of a group of elements consisting of ten decimal digits, the 26 lette the alphabet and a certain number of special symbols such as $. The t number of elements in an alphanumeric group is greater than 36. Therefo must be coded with minimum of six bits ($2^6 = 64$).

One possible arrangement of a six bit alphanumeric code is shown in t 1.4 under the name "internal code". The need to represent more tha characters gave rise to seven and eight bit alphanumeric codes. A such code is known as ASCII (American Standard Code for Informa Interchange), another is known as EBCDIC (Extended BCD Interchange C

**Table 1.4 Alphanumeric Character Codes**

| *Character* | *6-bit* | | *7-bit* | | *8-bit* | |
|---|---|---|---|---|---|---|
| | *Internal* | *Code* | *ASCII* | *Code* | *EBCDIC* | *Co* |
| A | 010 | 001 | 100 | 0001 | 1100 | 00 |
| B | 010 | 010 | 100 | 0010 | 1100 | 00 |
| C | 010 | 011 | 100 | 0011 | 1100 | 00 |
| D | 010 | 100 | 100 | 0100 | 1100 | 01 |
| E | 010 | 101 | 100 | 0101 | 1100 | 01 |
| F | 010 | 110 | 100 | 0110 | 1100 | 01 |
| G | 010 | 111 | 100 | 0111 | 1100 | 01 |
| H | 011 | 000 | 100 | 1000 | 1100 | 10 |
| I | 011 | 001 | 100 | 1001 | 1100 | 10 |
| J | 100 | 001 | 100 | 1001 | 1100 | 00 |
| K | 100 | 010 | 100 | 1010 | 1101 | 00 |
| L | 100 | 011 | 100 | 1011 | 1101 | 0011 |
| M | 100 | 100 | 100 | 1100 | 1101 | 0100 |
| N | 100 | 101 | 100 | 1101 | 1101 | 0101 |
| O | 100 | 110 | 100 | 1110 | 1101 | 0110 |
| P | 100 | 111 | 100 | 1111 | 1101 | 0111 |
| Q | 101 | 000 | 101 | 0000 | 1101 | 1000 |
| R | 101 | 001 | 101 | 0001 | 1101 | 1001 |
| S | 110 | 010 | 101 | 0010 | 1110 | 0010 |
| T | 110 | 011 | 101 | 0011 | 1110 | 0011 |
| U | 110 | 100 | 101 | 0100 | 1110 | 0100 |
| V | 110 | 101 | 101 | 0101 | 1110 | 0101 |
| W | 110 | 110 | 101 | 0110 | 1110 | 0110 |
| X | 110 | 111 | 101 | 0111 | 1110 | 0111 |
| Y | 111 | 000 | 101 | 1000 | 1110 | 1000 |
| Z | 111 | 001 | 101 | 1001 | 1110 | 1001 |
| 0 | 000 | 000 | 011 | 1010 | 1111 | 0000 |
| 1 | 000 | 001 | 011 | 0000 | 1111 | 0001 |
| 2 | 000 | 010 | 011 | 0001 | 1111 | 0010 |
| 3 | 000 | 011 | 011 | 0010 | 1111 | 0011 |
| 4 | 000 | 100 | 011 | 0011 | 1111 | 0100 |
| 5 | 000 | 101 | 011 | 0100 | 1111 | 0101 |
| 6 | 000 | 110 | 011 | 0101 | 1111 | 0110 |
| 7 | 000 | 111 | 011 | 0110 | 1111 | 0111 |
| 8 | 001 | 000 | 011 | 0111 | 1111 | 1000 |
| 9 | 001 | 001 | 011 | 1000 | 1111 | 1001 |
| Blank | 110 | 000 | 010 | 1001 | 0100 | 0000 |
| . | 011 | 011 | 010 | 0000 | 0100 | 1011 |
| ( | 111 | 100 | 010 | 1110 | 0100 | 1101 |
| + | 010 | 000 | 010 | 1000 | 0100 | 1110 |
| $ | 101 | 011 | 010 | 1011 | 0101 | 1011 |
| * | 101 | 100 | 010 | 0100 | 0101 | 1100 |
| ) | 011 | 100 | 010 | 1010 | 0101 | 1101 |
| – | 100 | 000 | 010 | 1001 | 0110 | 0000 |
| / | 110 | 001 | 010 | 1101 | 0110 | 0001 |
| , | 111 | 011 | 010 | 1111 | 0110 | 1011 |
| = | 001 | 011 | 011 | 1100 | 0111 | 1110 |

***Q.20. Explain Hamming and block codes.*** ***(R.G.P.V., June 2011)***

***Ans.*** **Hamming Code** – The Hamming code is a type of error correcting code. R.W. Hamming developed a system that provides a mathematical way to add one or more parity bits to a data character in order to detect and correct errors. The Hamming distance between two code words is defined as number of bits changed from one code word to another.

Consider $C_i$ and $C_j$ to be any two code words in a particular block code. Hamming distance $d_{ij}$ between the two vectors $C_i$ and $C_j$ is defined by the nu of components in which they differ. Assuming that $d_{ij}$ is determined for each of code words, the minimum value of the $d_{ij}$ can be called the Hamming dis $d_{min}$. For linear block codes, minimum weight is equal to minimum distance

For example,

$$\begin{array}{cccccccccc} C_i & = & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ & & \downarrow & & & \downarrow & \downarrow & & \\ C_j & = & 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{array}$$

Here, these code words differ in the leftmost bit position and in the fo and fifth bit positions from the left. Accordingly, $d_{ij} = d_{min} = 3$.

The 7-bit Hamming (7,4) code word $h_1\ h_2\ h_3\ h_4\ h_5\ h_6\ h_7$ associated a 4-bit binary number $b_3\ b_2\ b_1\ b_0$ is given as –

$$h_1 = b_3 \oplus b_2 \oplus b_0$$
$$h_2 = b_3 \oplus b_1 \oplus b_0,\ h_3 = b_3$$
$$h_4 = b_2 \oplus b_1 \oplus b_0,\ h_5 = b_2$$
$$h_6 = b_1,\ h_7 = b_1$$

where , $\oplus$ denotes the EX-OR operation. Note that bits $h_1$, $h_2$ and $h_4$ are e parity bits for the bit fields $b_3\ b_2\ b_0$, $b_3\ b_1\ b_0$ and $b_2\ b_1\ b_0$, respectively general, the parity bits ($h_1$, $h_2$, $h_4$, $h_8$, .....) are located in the positions co ponding to ascending powers of two (i.e., $2^0$, $2^1$, $2^2$, $2^3$, .... = 1, 2, 4, 8

The $h_1$ parity bit has a 1 in the LSB of its binary representation. There it checks all bit positions, including it, that have 1's in the same location ( LSB) in the binary representation (i.e., $h_1$, $h_3$, $h_5$ and $h_7$). The binary representa of $h_2$ has a 1 in the middle bit. Therefore, it checks all bit positions, includin that have 1's in the same location (i.e., middle bit) in the binary representa (i.e., $h_2$, $h_3$, $h_6$ and $h_7$). The binary representation of $h_4$ has a 1 in the M Therefore, it checks all bit positions, including it, that have 1's in the s location (i.e., MSB) in the binary representation (i.e., $h_4$, $h_5$, $h_6$ and $h_7$).

To decode a Hamming code, one must check for odd parity over th fields in which even parity was previously established. For example, a si bit error is indicated by a non-zero parity word $c_4\ c_2\ c_1$. where,

$$c_1 = h_1 \oplus h_3 \oplus h_5 \oplus h_7$$
$$c_2 = h_2 \oplus h_3 \oplus h_6 \oplus h_7$$
$$c_4 = h_4 \oplus h_5 \oplus h_6 \oplus h_7$$

If $c_4\ c_2\ c_1 = 000$, there is no error in the Hamming code. If it has a zero value, it indicates the bit position in error. For example, if $c_4\ c_2\ c_1$ = bit 5 is in error. To correct this error, bit 5 has to be complemented.

**Block Codes** – Block codes are also known as ***arithmetic codes***, or ***group codes***. In block codes, each block of k message bits is encoded into a block of n bits (n > k), as shown in fig. 1.1. The check bits are derived from the message bits and are added to them. The n-bit block of a channel encoder output is called a codeword and the codes or coding schemes in which the message bits appear at the beginning of a codeword, are called ***systematic codes***.

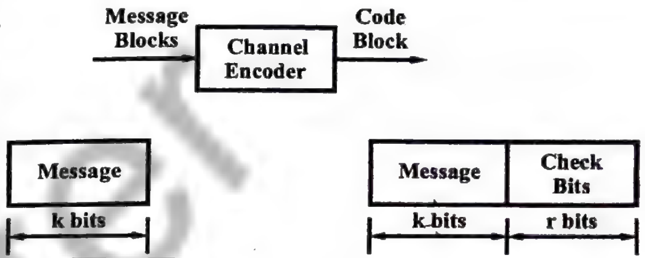


*Fig. 1.1*

Since there are k bits of information per block, there are $2^k$ possible distinct messages, out of the $2^n$ words that may be generated with n bits. This $2^k$ word set is called a ***block code .*** Every codeword has a code vector from the vector space $V_n$ of all n-tuples. Also, a linear code is defined as a set of $2^k$ n-tuples, which is a sub-space of the vector space $V_n$ of all n-tuples.

***Q.21. Write briefly about error detecting and error correcting codes.***
***(R.G.P.V., Dec. 2014)***

***Ans.*** **Error Detecting** – Various methods of error detecting are as follows –

**Parity** – The simplest technique for detecting errors is that of adding an extra bit, known as the parity bit, to each word being transmitted. There are two types of parity – odd parity and even parity. For odd parity, the parity bit is set to a 0 or a 1 at the transmitter such that the total number of 1 bits in the word including the parity bit is an odd number. For even parity, the parity bit is set to a 0 or 1 at the transmitter such that the total number of 1 bits in the word including the parity bit is an even number.

**Check-sums** – The parity method can detect only a single error within a word. If there are two errors within the same word, then simple parity will not detect them. A type of two-dimensional parity is used to overcome this disadvantage. Every transmitted word is added to the previously sent word and sum retained at the transmitter. The example is given below –

| | |
|---|---|
| Word A | 1101 0011 |
| Word B | 0001 1011 |
| Sum | 1110 1110 |

Each successive word is added to the previous sum in the same manner. The sum (called ***check sum***) is sent at the end of the transmission up to that time. The received check can be checked to the transmitted sum by the receiver. If both are the same, then there were no errors detected.

Check sum transmission is used in teleprocessing (TP) systems.

**Parity Data Codes** – Parity can be contained within each characte choice of codes carefully. Only two 1's are contained in each characte both the systems. The receiver can check this property in each rece character. The biquinary code is used in the abacus and 2 out of 5 co which is a unweighted code, is used in communication systems.

**Error Correcting Codes** – Refer the ans. of Q.20.

***Q.22. Write the difference between error-detecting code and er correcting code.*** ***(R.G.P.V., Nov./Dec. 2(***

*Ans.* During the process of data transmission, errors may occur. To de and correct, we use two types of codes, namely

(i) Error-detecting code (ii) Error-correcting code.

Error-correcting codes are more sophisticated than error detection co and require more redundancy bits. The number of bits required to corre multiple-bit or burst error is so high that in most cases it is inefficient to de For this reason, most error correction is limited to one, two, or three bit err

The Hamming code is a single-bit error correcting method using redund bits.

## NUMERICAL PROBLEMS

***Prob.21. Convert the following as directed –***

***(i)*** $(101111.101)_2 = (\ )_{10}$ ***(ii)*** $(10110101)_2 = (\ )_{Gray}$.

***(R.G.P.V., Dec. 20***

*Sol.* (i) $(101111.101)_2 = (1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times$
$+ (1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times$

$= 47 + 0.5 + 0.125 = \mathbf{(47.625)_{10}}$ A

(ii) $(10110101)_2 = (\ )_{Gray}$

To convert the given binary number into Gray, the procedure as follo

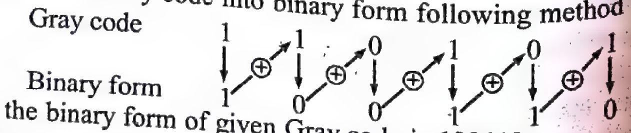
$$\begin{array}{cccccccc} 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ \hline 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ \hline \end{array}$$

$(10110101)_2 = \mathbf{(11101111)_{Gray}}$ A

***Prob.22. Convert the Gray code 110101 to binary form.***

***(R.G.P.V., Nov./Dec. 2007, Feb. 2***

*Sol.* To convert Gray code into binary form following method is us



Hence, the binary form of given Gray code is **100110**

***Prob.23. Convert 10111011 in binary into its equivalent gray code.*** ***(R.G.P.V., June 2015)***

*Sol.* Given, $(10111011)_2 = (\ )_{Gray}$

To convert the given binary number into Gray, the procedure as follows –

$$\begin{array}{ccccccccccccccc} 1 & \oplus & 0 & \oplus & 1 & \oplus & 1 & \oplus & 1 & \oplus & 0 & \oplus & 1 & \oplus & 1 \\ \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow \\ 1 & & 1 & & 1 & & 0 & & 0 & & 1 & & 1 & & 0 \end{array}$$

Hence, $(10111011)_2 = \mathbf{(11100110)_{Gray}}$ **Ans.**

***Prob.24. Express decimal 5280 in excess-3 code. (R.G.P.V., Dec. 2010)***

*Sol.* Given decimal number 5 2 8 0

$$\begin{array}{lcccc} \text{Add 3 to each bit} = & 5 & 2 & 8 & 0 \\ & +3 & +3 & +3 & +3 \\ \hline \text{Sum} \rightarrow & 8 & 5 & 11 & 3 \\ & \downarrow & \downarrow & \downarrow & \downarrow \\ \text{Binary} \rightarrow & 1000 & 0101 & 1011 & 0011 \end{array}$$

Thus,

$(5280)_{10} = \mathbf{(1000\ 0101\ 1011\ 0011)_{EX\text{-}3}}$ **Ans.**

***Prob.25. Convert the following codes as directed –***

***(i)*** $(785.B2)_{16} = (\ )_{10}$ ***(ii)*** $(1011011.1101)_2 = (\ )_8$

***(iii)*** $(110101.101101)_2 = (\ )_{Gray}$ ***(iv)*** $(751.231)_8 = (\ )_{16}$.

***(R.G.P.V., Dec. 2012)***

*Sol.* *(i)* $(785.B2)_{16} = 7 \times 16^2 + 8 \times 16^1 + 5 \times 16^0 + B \times 16^{-1} + 2 \times 16^{-2}$

$$= 1792 + 128 + 5 + \frac{11}{16} + \frac{2}{256}$$

$= \mathbf{(1925.695313)_{10}}$ **Ans.**

*(ii)* $(1011011.1101)_2 = (\ )_8$

$= \underbrace{001}_{1}\ \underbrace{011}_{3}\ \underbrace{011}_{3}.\underbrace{110}_{6}\ \underbrace{100}_{4} = \mathbf{(133.64)_8}$ **Ans.**

*(iii)* $(110101.101101)_2 = (\ )_{Gray}$

The binary number to gray code formation is obtained as –

$$\begin{array}{ccccccccccccccccccccccc} 1 & \oplus & 1 & \oplus & 0 & \oplus & 1 & \oplus & 0 & \oplus & 1 & \oplus & 1 & \oplus & 0 & \oplus & 1 & \oplus & 1 & \oplus & 0 & \oplus & 1 \\ \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow \\ 1 & & 0 & & 1 & & 1 & & 1 & & 1 & & 0 & & 1 & & 1 & & 0 & & 1 & & 1 \end{array}$$

$= \mathbf{(101111.011011)_{Gray}}$ **Ans.**

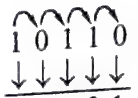
*(iv)* $(751.231)_8 = \underbrace{0001}_{1}\ \underbrace{1110}_{E}\ \underbrace{1001}_{9}\cdot\underbrace{0100}_{4}\ \underbrace{1100}_{C}\ \underbrace{1000}_{8}$

$= \mathbf{(1E9.4C8)_{16}}$ **Ans.**

***Prob.26. (i) Convert binary $(10110)_2$ to Gray code.***

***(ii) The seven bit Hamming code as received is 0010001. Assum that even parity has been used, check is it correct ? If not find correct co***

***(R.G.P.V., June 20***



***Sol.*** (i) $(10110)_2 = 1\ 1\ 1\ 0\ 1$

$(10110)_2 = \mathbf{(11101)}_{\text{Gray}}$

(ii)

| $d_7$ | $d_6$ | $d_5$ | $r_4$ | $d_3$ | $r_2$ | $r_1$ |
|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 | 1 |

Now, $r_1$ will take care of $r_1$, $d_3$, $d_5$, $d_7$ whose values are 1010 i.e., t 1's. Even parity is satisfied. Hence $r_1 = 0$.

$r_2$ will take care of $r_2$, $d_3$, $d_6$, $d_7$ whose values are 0000 i.e., No l Hence $r_2 = 0$.

$r_4$ will take care of $r_4$, $d_5$, $d_6$ and $d_7$, whose values are 0100 i.e. one l Hence $r_4 = 1$.

Hence error word = $(r_4, r_2, r_1) = (1\ 0\ 0) = (4)_{10}$

Hence the bit 4 of the transmission is in error. Since $r_4$ is received as therefore correct codeword is –

= **0011001**

## LOGIC GATES

***Q.23. What are logic gates ?*** ***(R.G.P.V., June 20***

***Ans.*** The logic gates are electronic circuits because they are made up a number of electronic devices and components. The logic circuits that perfo the logical operations of AND, OR and NOT are called gates.

The inputs and outputs of logic gates can occur only in two levels, s as HIGH (or 1) and LOW (or 0). The table that lists all the possible combinati of input variables and the corresponding outputs is known as truth table. T logic gates are the building blocks of hardware, which are available in form of various IC families. Each gate has a distinct logic symbol and operation can be described by means of an algebraic function. The level lo is defined as a logic in which the voltage levels represent logic-1 and logi Level logic may be positive logic or negative logic.

***Q.24. Describe the operation of NOT (inverter) logic gate using tr table and standard logic symbols.***

***Ans.*** The inverter circuit performs the operation called complementat The inverter changes one logic level to the opposite logic level. It has one in and one output. When a HIGH level is applied to an inverter, a LOW level appears at its output. The standard logic symbol of NOT gate is shown in figs. 1.2 (a) and (b), respectively.

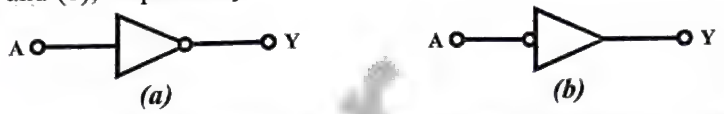


***Fig. 1.2 Standard Logic Symbols of NOT Gate***

The logic equation of NOT gate is written as –

$$Y = \overline{A}$$

This expression read as Y equals complement of A. The truth table of a NOT gate is given in table 1.5.

The output of NOT gate is always complement of its input. The presence of a small circle called the bubble, always indicates inversion in digital circuits.

**Table 1.5 Truth Table**

| *Input* | *Output* |
|---|---|
| *A* | *Y = A* |
| 0 | 1 |
| 1 | 0 |

***Q.25. Explain the operation of AND gate using truth table and standard logic symbol.***

***Ans.*** An AND gate performs logical multiplication, which is commonly known as AND function. The output of AND gate is 1 if and only if all the inputs are 1. If A and B are two input variables of an AND gate and Y is its output, then we have –

$$Y = A.B = AB$$

The standard logic symbol of AND gate is shown in fig. 1.3.

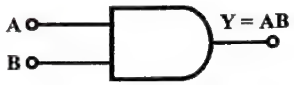


***Fig. 1.3 Standard Logic Symbol of Two Input AND Gate***

**Table 1.6 Truth Table**

| *Inputs* | | *Output* |
|---|---|---|
| *A* | *B* | *Y = AB* |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

The truth table of two input AND gate is given in table 1.6. The logical equation Y = AB is read as Y equals A and B, which means that Y will be 1 only when A and B are both 1.

***Q.26. Describe the operation of OR gate using truth table and standard logic symbol.***

***Ans.*** An OR gate has two or more inputs and performs logical addition. An OR gate produces a 1 on the output when any of the inputs is 1. The output is 0 only when all of the inputs are 0. If A and B are the two input variables of an OR gate and Y is its output, then we get

$$Y = A + B$$

The logical equation Y = A + B is read as Y equals A OR B or A plus B. T standard logic symbol of OR gate is given in fig. 1.4.

The truth table of two input OR gate is given in table 1.7.

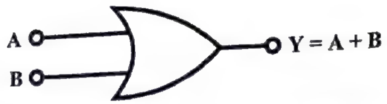


Fig. 1.4 Standard Logic Symbol of OR Gate

**Table 1.7 Truth Table**

| *Inputs* | | *Output* |
|---|---|---|
| *A* | *B* | *Y = A + B* |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

***Q.27. How will you construct the following gates from a diode-resis network –***

***(i) AND (ii) OR (iii) NOT.***

***(R.G.P.V., Dec. 20***

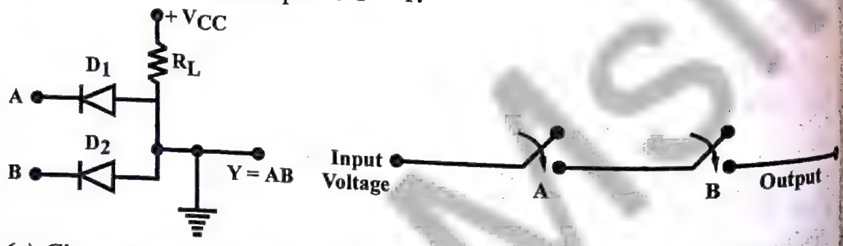
***Ans.(i) AND Gate*** – A 2-inputs AND gate using diodes is shown in f 1.5, in which A and B represent the inputs and Y the output.

If A = 0 and B = 0, both the diodes conduct as they are forward bias and hence the output is Y = 0.

If A = 0 and B = 1, the diode $D_1$ conducts and $D_2$ does not conduct, hence the output is Y = 0.

If A = 1 and B = 0, the diode $D_1$ does not conduct and $D_2$ conducts, hence the output is Y = 0.

If A = 1 and B = 1, both the diodes do not conducts as they are rever biased, and hence the output is Y = 1.



*(a) Circuit Diagram using Diodes* *(b) Its Electrical Equivalent*

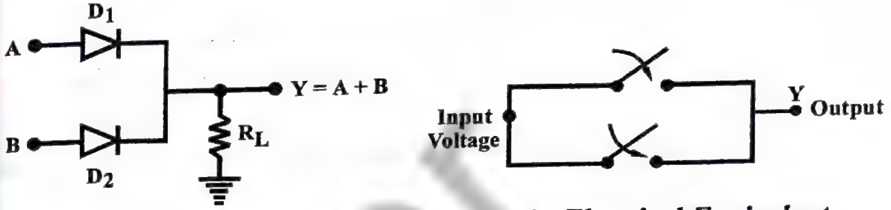
*Fig. 1.5 2-input AND Gate*

***(ii) OR Gate*** – An OR gate using diodes is shown in fig. 1.6, which A and B represent the inputs and Y the output. The resistance $R_L$ is load resistance.

If A = 0 and B = 0 both the diodes will not conduct and hence the out Y = 0.

If A = 1 and B = 0, diode $D_1$ conducts, then $V_0 \approx 5V$ and so Y = 1
If A = 0 and B = 1, diode $D_2$ conducts and hence Y = 1

If A = 1 and B = 1, both the diodes conduct and hence Y = 1.



*(a) Circuit Diagram using Diodes* *(b) Its Electrical Equivalent*

*Fig. 1.6 2-input OR Gate*

***(iii) NOT Gate*** – A NOT gate using a transistor is shown in fig. 1.7, in which A represents the input and Y represent the output i.e., Y = $\overline{A}$. When the input is HIGH, the transistor is in the ON state and the output $V_C$ = $V_{CE(sat)}$ is LOW. If the input is LOW, the transistor is in the OFF state and the output $V_C = V_{CC}$ is HIGH.

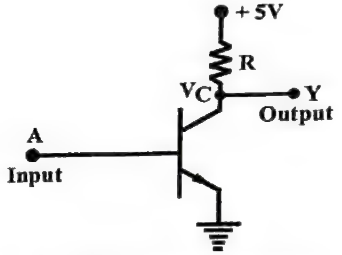


*Fig. 1.7 Circuit Diagram of NOT Gate using Transistor*

***Q.28. Define NAND and NOR gates and give their truth tables. Write down the boolean expressions for the output of each gate.***

***(R.G.P.V., Nov. 2018)***

***Ans.* NAND Gate** – The term NAND is a combination of the NOT-AND gates and implies an AND function with a complemented output. It has two or more inputs and only one output. A NAND gate produces a 0 (or LOW) output only when all the inputs are 1 (or HIGH). When any one of input is 0 (or LOW), the output will be 1 (or HIGH). Here we note that this operation is opposite that of the AND in terms of the output level. The logic symbol for the NAND gate is shown in fig. 1.8.

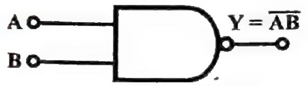


*Fig. 1.8 Logic Symbol of NAND Gate*

**Table 1.8 Truth Table**

| *Inputs* | | *Output* |
|---|---|---|
| *A* | *B* | *Y = $\overline{AB}$* |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

The truth table of two input NAND gate is given in table 1.8.

The logical operation of a NAND gate is represented as –

$$Y = \overline{AB}$$

The logical equation Y = A + B is read as Y equals A OR B or A plus B. standard logic symbol of OR gate is given in fig. 1.4.

The truth table of two input OR gate is given in table 1.7.

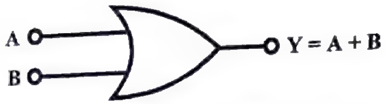


*Fig. 1.4 Standard Logic Symbol of OR Gate*

**Table 1.7 Truth Table**

| *Inputs* | | *Output* |
|---|---|---|
| *A* | *B* | *Y = A+B* |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

***Q.27. How will you construct the following gates from a diode-resi network –***

***(i) AND (ii) OR (iii) NOT.***

***(R.G.P.V., Dec. 2***

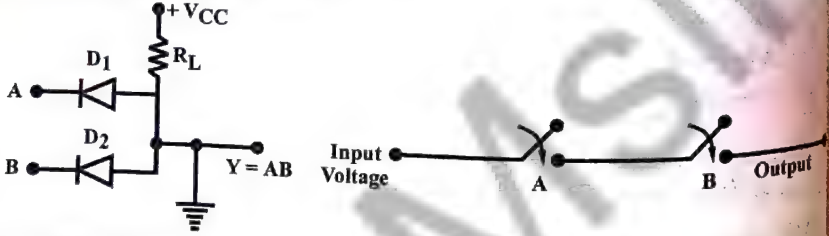
***Ans.(i) AND Gate*** – A 2-inputs AND gate using diodes is shown in 1.5, in which A and B represent the inputs and Y the output.

If A = 0 and B = 0, both the diodes conduct as they are forward bia and hence the output is Y = 0.

If A = 0 and B = 1, the diode $D_1$ conducts and $D_2$ does not conduct, hence the output is Y = 0.

If A = 1 and B = 0, the diode $D_1$ does not conduct and $D_2$ conducts, hence the output is Y = 0.

If A = 1 and B = 1, both the diodes do not conducts as they are reve biased, and hence the output is Y = 1.



***(a) Circuit Diagram using Diodes*** ***(b) Its Electrical Equivalent***

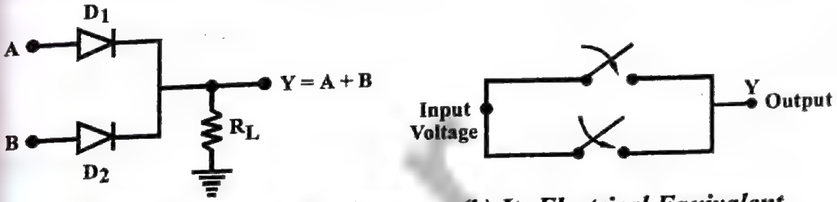
***Fig. 1.5 2-input AND Gate***

***(ii) OR Gate*** – An OR gate using diodes is shown in fig. 1.6, which A and B represent the inputs and Y the output. The resistance $R_L$ is load resistance.

If A = 0 and B = 0 both the diodes will not conduct and hence the ou Y = 0.

If A = 1 and B = 0, diode $D_1$ conducts, then $V_0 \approx 5V$ and so Y = 1
If A = 0 and B = 1, diode $D_2$ conducts and hence Y = 1

If A = 1 and B = 1, both the diodes conduct and hence Y = 1.



***(a) Circuit Diagram using Diodes*** ***(b) Its Electrical Equivalent***

***Fig. 1.6 2-input OR Gate***

***(iii) NOT Gate*** – A NOT gate using a transistor is shown in fig. 1.7, in which A represents the input and Y represent the output i.e., Y = $\overline{A}$. When the input is HIGH, the transistor is in the ON state and the output $V_C$ = $V_{CE(sat)}$ is LOW. If the input is LOW, the transistor is in the OFF state and the output $V_C = V_{CC}$ is HIGH.

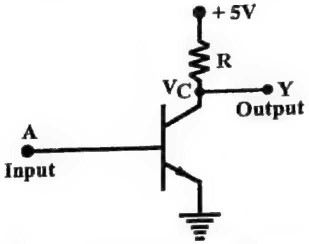


***Fig. 1.7 Circuit Diagram of NOT Gate using Transistor***

***Q.28. Define NAND and NOR gates and give their truth tables. Write down the boolean expressions for the output of each gate.***

***(R.G.P.V., Nov. 2018)***

***Ans.* NAND Gate** – The term NAND is a combination of the NOT-AND gates and implies an AND function with a complemented output. It has two or more inputs and only one output. A NAND gate produces a 0 (or LOW) output only when all the inputs are 1 (or HIGH). When any one of input is 0 (or LOW), the output will be 1 (or HIGH). Here we note that this operation is opposite that of the AND in terms of the output level. The logic symbol for the NAND gate is shown in fig. 1.8.

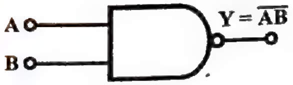


***Fig. 1.8 Logic Symbol of NAND Gate***

**Table 1.8 Truth Table**

| *Inputs* | | *Output* |
|---|---|---|
| *A* | *B* | *Y = $\overline{AB}$* |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

The truth table of two input NAND gate is given in table 1.8.

The logical operation of a NAND gate is represented as –

$$Y = \overline{AB}$$

**NOR Gate** – NOR is a combination of NOT-OR gates. A NOR g produces a 0 (or LOW) output when any of its input is 1 (or HIGH) produces output 1 (or HIGH) only when all of its inputs are 0 (or LOW). logical equation of the two input NOR gate is given as –

$$Y = \overline{A + B}$$

The standard logic symbol of NOR gate is given in fig. 1.9 and its t table is shown in table 1.9.

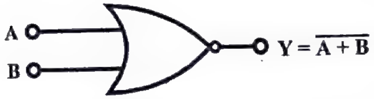


*Fig. 1.9 Standard Logic Symbol of NOR Gate*

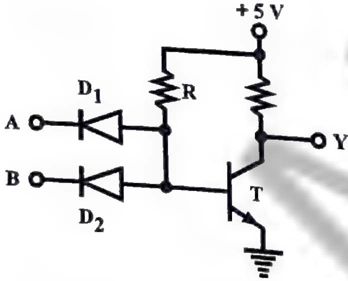
**Table 1.9 Truth Table**

| *Inputs* | | *Output* |
|---|---|---|
| *A* | *B* | *Y* = $\overline{A+B}$ |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

***Q.29. Realize NAND, NOR and NOT gates using transistors.***

***(R.G.P.V., May 20***

***Ans.*** **NAND Gate using Transistors** – Fig. 1.10 (a) shows a disc two-input NAND gate. The diodes $D_1$ and $D_2$ are OFF when A = + 5 V an = + 5 V. The transistor T obtains sufficient base drive from the supply vi Thus the transistor T is ON and the output is equal to zero (i.e., Y = $V_{ce(sa}$ 0 V). The transistor T is OFF when A = 0 V or B = 0 V or when both A an are equal to zero. Thus the output is equal to + 5 V. Fig. 1.10 (b) shows truth table.



*(a) Discrete NAND Gate*

Truth Table

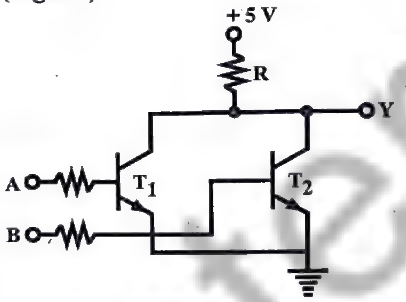
| Inputs | | Output |
|---|---|---|
| A | B | Y |
| 0 V | 0 V | 5 V |
| 0 V | 5 V | 5 V |
| 5 V | 0 V | 5 V |
| 5 V | 5 V | 0 V |

*(b)*

*Fig. 1.10*

**NOR Gate using Transistors** – Fig. 1.11 (a) and (b) shows the t input NOR gate and truth table. Transistors $T_1$ and $T_2$ are OFF when A = ( and B = 0 V. So, no current flows via R and thus no voltage drop takes p across R. Therefore, the output voltage is equal to + 5 V (logic 1). If ei

A = + 5 V or B = + 5 V or if both A and B are equal to + 5 V, the corresponding transistor $T_1$ or $T_2$ or both $T_1$ and $T_2$ are ON. Thus the output voltage is equal to 0 V (logic 0).



*(a) Discrete NOR Gate*

Truth Table

| Inputs | | Output |
|---|---|---|
| A | B | Y |
| 0 V | 0 V | 5 V |
| 0 V | 5 V | 0 V |
| 5 V | 0 V | 0 V |
| 5 V | 5 V | 0 V |

*(b)*

*Fig. 1.11*

**NOT Gate using Transistor** – Refer the ans. of Q.27 (iii).

***Q.30. Discuss AND, OR, X-OR and X-NOR gates. (R.G.P.V., June 2011)***

***Ans.*** **AND Gates** – Refer the ans. of Q.25.

**OR Gates** – Refer the ans. of Q.26.

**X-OR Gates** – The EX-OR gate is recognizes only words that have an odd number of 1's. It has two or more input and one output. The output of a two input EX-OR gate assumes a HIGH (or 1) if one and only one input assumes a HIGH (or 1). Because of their fundamental importance in many applications, these gates are often used as basic logic elements with their own unique symbols. The standard logic symbol for an EX-OR gate is given in fig. 1.12 and the truth table for its operation is shown in table 1.10.

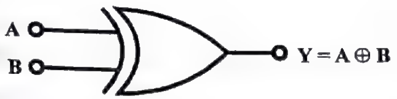


*Fig. 1.12 Standard Logic Symbol of EX-OR Gate*

**Table 1.10 Truth Table**

| *Inputs* | | *Output* |
|---|---|---|
| *A* | *B* | $Y = \overline{A}B + A\overline{B}$<br>$= A \oplus B$ |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

The logical equation of two input EX-OR gate is given as –

$$Y = \overline{A}B + A\overline{B} = A \oplus B$$

This expression is read as Y equals A EX-OR B.

**X-NOR Gates** – The EX-NOR gate is logically equivalent to an EX-OR gate followed by an inverter (or NOT gate). The bubble on the output of the

EX-NOR symbol denotes that its output is opposite that of EX-OR gate. EX-NOR gate has two or more inputs and one output. The output of a input EX-NOR gate is HIGH if both the inputs A and B are HIGH or LOW. output is LOW if the one input is LOW and other input is HIGH.

The standard logic symbol of EX-NOR gate is shown in fig. 1.13 and truth table for operation is given in table 1.11.

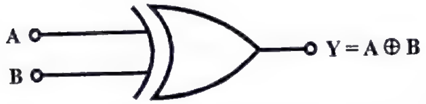


*Fig. 1.13 Standard Logic Symbol of EX-OR Gate*

**Table 1.11 Truth Table**

| *Inputs* | | *Output* |
|---|---|---|
| *A* | *B* | $Y = (\overline{\overline{A}B + A\overline{B}}) = (\overline{A \oplus B})$ |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

***Q.31. Draw the logic diagram of EX-NOR gate using only NOR ga***
***(R.G.P.V., Dec. 20***

***Ans.*** The output of EX-NOR gate is given by

$$Y = \overline{A \oplus B} = \overline{\overline{A}B + A\overline{B}} = \overline{\overline{A}B}.\overline{A\overline{B}} = (\overline{\overline{A}} + \overline{B}).(\overline{A} + \overline{\overline{B}}) = (A + \overline{B})(\overline{A} +$$

The logic diagram of EX-NOR gate is shown in fig. 1.14.

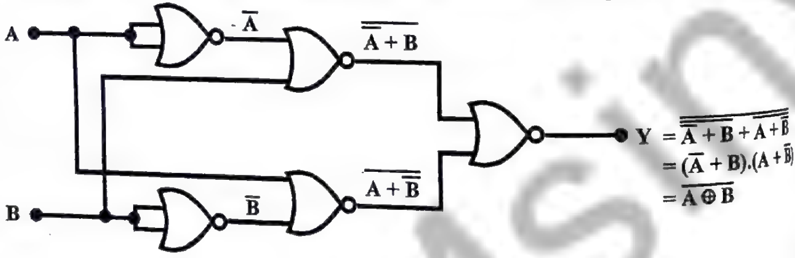

A
B
$\overline{A}$
$\overline{\overline{A}+B}$
$\overline{B}$
$\overline{A+\overline{B}}$
$Y = \overline{\overline{\overline{A}+B}+\overline{A+\overline{B}}}$
$= (\overline{A}+B).(A+\overline{B})$
$= \overline{A \oplus B}$

*Fig. 1.14*

***Q.32. Compare truth table, excitation table and state table.***
***(R.G.P.V., Dec. 2006, June 20***

***Ans.*** The comparison between truth table, excitation table and state ta are as follows –

| *S.No.* | *Truth Table* | *Excitation Table* | *State Table* |
|---|---|---|---|
| (i) | It gives all possible combinations of input variables and the corresponding output. | It gives information about the excitation or input required to be applied to the memory element. | It is a tabular repre ntation of relations between the prese state, the input, the n state, and the outp |
| (ii) | It shows how the logic circuit's output responds to various combination of logic level at the inputs. | It shows the information about the output of the machine after the application of the present input. | It shows the output that will be generated and the next state to which the machine will go. |

***Q.33. What do you understand by universal gate ? Design all logic gates using universal gates.*** ***(R.G.P.V., June 2009)***

***Or***

***What is universal gate ?*** ***(R.G.P.V., June 2014)***

***Or***

***What are universal gates ? Explain with example. (R.G.P.V., Dec. 2014)***

***Or***

***What are universal gates ? Why are they called so ? (R.G.P.V., Dec. 2015)***

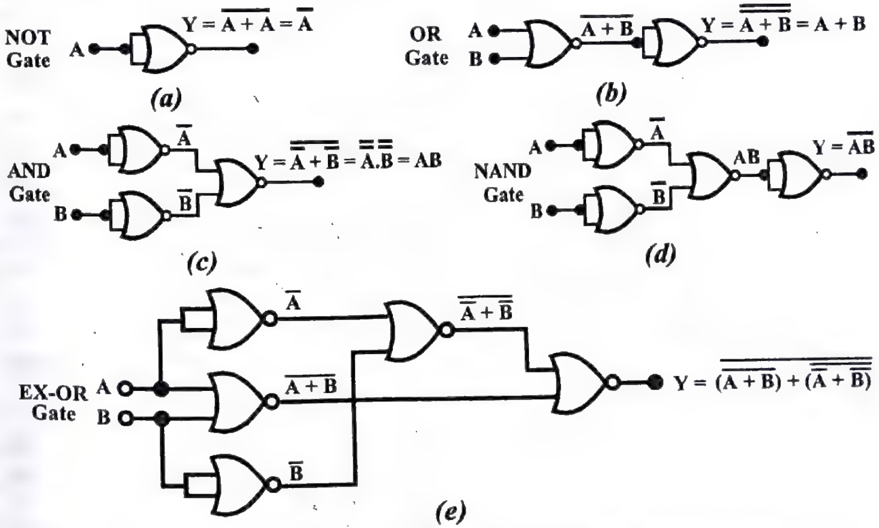
***Or***

***Why NAND gate is known as universal gate ?*** ***(R.G.P.V., Dec. 2016)***

***Or***

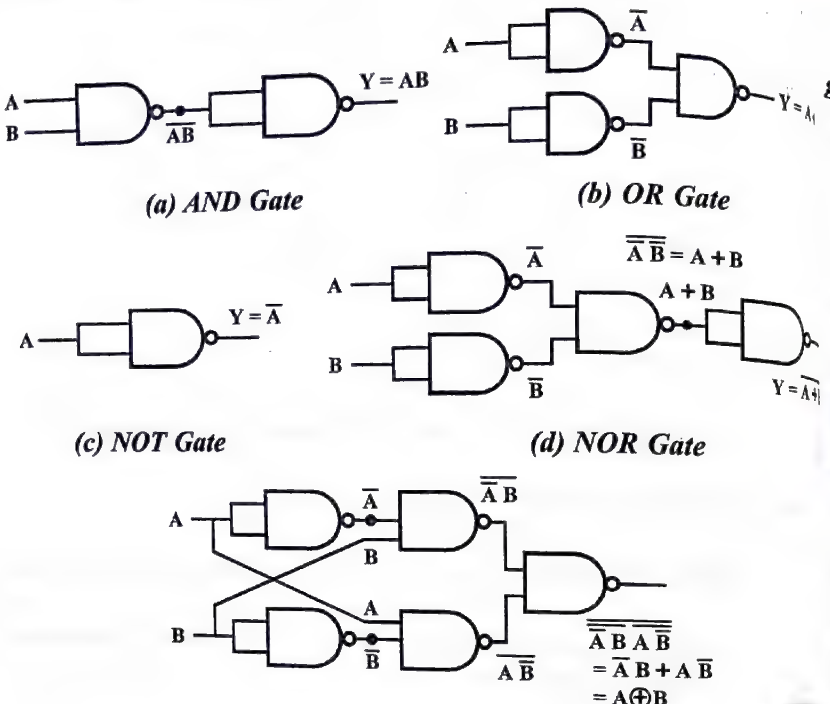
***What is universal gate ? Implement AND, OR and NOT gates using NAND gates and NOR gates.*** ***(R.G.P.V., Dec. 2017)***

***Ans.*** NAND and NOR gates are called universal gates or universal building blocks because both can be used to implement any gate like AND, OR and NOT gates or any combination of these basic gates. Fig. 1.15 shows how a NOR gate can be used to realize various logic gates while fig. 1.16 shows how a NAND gate can be used for the same.


NOT Gate
A
$Y = \overline{A+A} = \overline{A}$
OR Gate
A
B
$\overline{A+B}$
$Y = \overline{\overline{A+B}} = A+B$
AND Gate
A
B
$\overline{A}$
$\overline{B}$
$Y = \overline{\overline{A}+\overline{B}} = \overline{\overline{A}}.\overline{\overline{B}} = AB$
NAND Gate
A
B
$\overline{A}$
$\overline{B}$
AB
$Y = \overline{AB}$
EX-OR Gate
A
B
$\overline{A}$
$\overline{A+B}$
$\overline{B}$
$\overline{\overline{A}+\overline{B}}$
$Y = \overline{(\overline{A+B}) + (\overline{\overline{A}+\overline{B}})}$

*(a)* *(b)* *(c)* *(d)* *(e)*

***Fig. 1.15 Realization of Various Gates using NOR Gate***


A
B
$\overline{AB}$
Y = AB
A
B
$\overline{A}$
$\overline{B}$
Y = A
A
Y = $\overline{A}$
A
B
$\overline{A}$
$\overline{B}$
$\overline{\overline{A}\,\overline{B}}$ = A + B
A + B
Y = $\overline{A+}$
A
B
$\overline{A}$
B
A
$\overline{B}$
$\overline{\overline{A}B}$
$\overline{A\overline{B}}$
$\overline{\overline{\overline{A}B}\;\overline{A\overline{B}}}$
= $\overline{A}$B + A$\overline{B}$
= A⊕B

(a) AND Gate

(b) OR Gate

(c) NOT Gate

(d) NOR Gate

(e) EX-OR Gate

*Fig. 1.16 Realization of Various Gates using NAND Gate*

***Q.34. Implement EX-OR gate using NOR gates.*** ***(R.G.P.V., June 201***

***Ans.*** Implement EX-OR gate using NOR gates is shown in fig. 1.15(

$$
\begin{aligned}
Y &= A\overline{B}+\overline{A}B \\
&= A\overline{A}+A\overline{B}+\overline{A}B+B\overline{B} \qquad [A\overline{A}= \\
&= A(\overline{A}+\overline{B})+B(\overline{A}+\overline{B}) = (A+B)(\overline{A}+\overline{B}) \\
&= \overline{\overline{(A+B)(\overline{A}+\overline{B})}} = \overline{\overline{(A+B)}+\overline{(\overline{A}+B)}}
\end{aligned}
$$

## NUMERICAL PROBLEMS

***Prob.27. Implement y = AB + CD using only NAND gates.*** ***(R.G.P.V., June 201***

***Sol.*** Given function –

$$y = AB + CD$$

The implementation of the given function is shown in fig. 1.17.

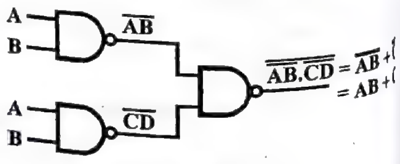

A
B
$\overline{AB}$
A
B
$\overline{CD}$
$\overline{\overline{AB}.\overline{CD}}$ = $\overline{\overline{AB}}$+
= AB+

*Fig. 1.17*

***Prob.28. Implement the function F = A(B + CD) + BC' using NOR gate.*** ***(R.G.P.V., June 2017)***

***Sol.*** Implementation of given function using NOR gate is shown in fig. 1.18.

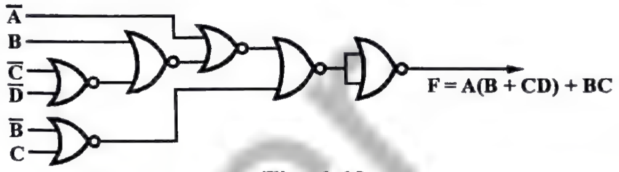


*Fig. 1.18*

## BOOLEAN ALGEBRA, BOOLEAN FUNCTIONS, SIMPLIFICATION OF BOOLEAN FUNCTIONS

***Q.35. Write short note on Boolean algebra.***

***Ans.*** A number system based on two digits 0 and 1 is known as binary number system. For manipulations of binary variables an english mathematician George Boolean developed laws, known as Boolean algebra. The Boolean algebra differs from both ordinary algebra and the binary number system. For example, in Boolean algebra the addition of two digits 1 + 1 = 1, while in binary arithmetic this results in 10.

***Q.36. State duality theorem.*** ***(R.G.P.V., May/June 2006)***

***Ans.*** The Huntington postulates have been listed in pair and designated by part (a) and part (b). One part may be obtained from the other if binary operator and the identity element are interchanged. This important property of Boolean algebra is called ***duality principle*** or ***duality theorem.*** It states that every algebraic expression deducible from the postulates of Boolean algebra remain valid if the operator and identity element are interchanged. In a two valued Boolean algebra, the identity element and the element of set B are same, i.e., 1 and 0. The duality principle have many applications. If the dual of an algebra expression is desired, we simply change the AND and OR operator and replace 1's by 0's and 0's by 1's.

***Q.37. State and prove basic laws of Boolean algebra.***
***(R.G.P.V., Dec. 2013)***

***Ans.*** There are fundamental laws in Boolean algebra, which are used to build a workable, cohesive framework upon which are placed the theorems proceeding from these laws. These laws are as follows –

**Laws of Complementation** – The term complement simply means to invert or to change 1's to 0's and 0's to 1's. The five laws of complementation are given below –

Law 1 — $\overline{0} = 1$

Law 2 — $\overline{1} = 0$

Law 3 — If $A = 0$, then $\overline{A} = 1$

Law 4 — If $A = 1$, then $\overline{A} = 0$

Law 5 — $\overline{\overline{A}} = A$

**AND Laws** – There are four AND laws as given below –

Law 1 — $A \,.\, 0 = 0$

Law 2 — $A \,.\, 1 = A$

Law 3 — $A.A = A$

Law 4 — $A \,.\, \overline{A} = 0$

**OR Laws** – The four OR laws are as follows –

Law 1 — $A + 0 = A$

Law 2 — $A + 1 = 1$

Law 3 — $A + A = A$

Law 4 — $A + \overline{A} = 1$

**Commutative Laws** – The commutative laws allow the change in posi of an AND or an OR variable –

Law 1 – $A + B = B + A$

Law 2 – $A \,.\, B = B \,.\, A$

**Associative Laws** – The associative laws allow the grouping of variabl There are two associative laws –

Law 1 – $A + (B + C) = (A + B) + C$

Law 2 – $A \,.\, (B \,.\, C) = (A \,.\, B) \,.\, C$

**Distributive Laws** – The distributive laws allow the factoring multiplying out of expressions. Three distributive laws are as follows –

Law 1 – $A + BC = (A + B)\,(A + C)$

This law states that the AND operation of several variables and the operation of the result with a single variable is equivalent to the OR operat of the single variable with each of the several variables and then the A operation of the sums.

Law 2 – $A(B + C) = AB + AC$

This law states that the OR operation of several variables and the A operation of the result with a single variable is equivalent to the AND operat of the single variable with each of the several variables and then the operation of the products.

Law 3 – $A + \overline{A}B + A + B$

This law states that the OR operation of a variable with the AND of complement of that variable with another variable, is equal to the OR operat of the two variables.

**Absorption Laws** – There are two laws of absorption which are as follows –

Law 1 — $A + A \,.\, B = A$

The law states that ORing of a variable, A with the AND of that variable, A and another variable, B is equal to that variable itself.

It is expressed algebraically as –

$$A + A \,.\, B = A\,(1 + B) = A.1 = A$$

Law 2 — $A\,(A + B) = A$

The law states that ANDing of a variable, A with the OR of variable, A and variable, B is equal to that variable itself. It is illustrates algebraically as –

$$A\,(A + B) = A.A + A.B = A + A.B = A\,(1 + B) = A.1 = A$$

***Q.38. Write five theorem of Boolean algebra and simplify***

***$F = (A + B)'\,(A' + B')'$.*** ***(R.G.P.V., May 2018)***

***Ans.*** Refer the ans. of Q.37.

Given,

$$\begin{aligned} F &= (A + B)'\,(A' + B')' \\ &= (A'.B').(A.B) \\ &= A\,A'\,B\,B' \\ &= 0 \qquad\qquad (\because A.A' = 0) \end{aligned}$$

***Q.39. What is Boolean algebra write any three theorems of Boolean algebra ?*** ***(R.G.P.V., Dec. 2017)***

***Ans.*** Refer the ans. of Q.35 and Q.37.

***Q.40. State the distributive property of Boolean algebra.*** ***(R.G.P.V., Dec. 2016)***

***Ans.*** Refer the ans. of Q.37, under the heading Distributive law.

***Q.41. What is minterm ? Write all minterm for three variables.*** ***(R.G.P.V., Nov./Dec. 2007)***

***Ans.*** A product term containing all the K-variables of the function in either complemented or uncomplemented form is known as a ***minterm***. A 2-variables function has four possible combinations, which are $\overline{A}\,\overline{B}$, $\overline{A}B$, $A\overline{B}$ and $AB$. These product terms are referred to as minterms or standard product or fundamental products. There are 8 minterms for a 3-binary input variables function, as shown in table 1.12. Each minterm can be

**Table 1.12 Minterm Table**

| *Inputs* | | | *Minterm* | *Designation* |
|---|---|---|---|---|
| *A* | *B* | *C* | | |
| 0 | 0 | 0 | $\overline{A}\,\overline{B}\,\overline{C}$ | $m_0$ |
| 0 | 0 | 1 | $\overline{A}\,\overline{B}\,C$ | $m_1$ |
| 0 | 1 | 0 | $\overline{A}\,B\,\overline{C}$ | $m_2$ |
| 0 | 1 | 1 | $\overline{A}\,B\,C$ | $m_3$ |
| 1 | 0 | 0 | $A\,\overline{B}\,\overline{C}$ | $m_4$ |
| 1 | 0 | 1 | $A\,\overline{B}\,C$ | $m_5$ |
| 1 | 1 | 0 | $A\,B\,\overline{C}$ | $m_6$ |
| 1 | 1 | 1 | $A\,B\,C$ | $m_7$ |

achieved by the AND operation of all the variables of the function. In a minterm, a variable appears either in uncomplemented form, if it has a valu 1 in the corresponding combination, or in complemented form, if it posses the value of 0. The minterms of a 3-variable function can be denoted by $m_1$, $m_2$, $m_3$, $m_4$, $m_5$, $m_6$ and $m_7$.

An important property of a minterm is that it contains the value 1 for o one combination of K input variables; i.e., for a K-variable function of the minterms, only one minterm will possess the value 1, while the remaining $2^K$ minterms will have the value 0 for an arbitrary input combination.

***Q.42. Write short note on maxterm.***

***Ans.*** A sum term containing all the K-variables of the function in eit complemented or uncomplemented form is referred to as a ***maxterm***. A tw variable function has four possible combinations which are A + B, A + $\overline{A}+B$ and $\overline{A}+\overline{B}$. These sums are known as ***maxterms***. Therefore, a binary input variable function has 8 maxterms as given in table 1.13. Each maxterm can be achieved by the OR operation of all the variables of the function. In a maxterm, a variable appears either in uncomplemented form if it contains the value 0 in the corresponding combination or in complemented form if it possesses the value 1. In a 3-variable function, the maxterms can be represented by $M_0$, $M_1$, $M_2$, $M_3$, $M_4$, $M_5$, $M_6$ and $M_7$.

**Table 1.13 Maxterm Table**

| *Inputs* | | | *Maxterm* | *Designatio* |
|---|---|---|---|---|
| *A* | *B* | *C* | | |
| 0 | 0 | 0 | $A+B+C$ | $M_0$ |
| 0 | 0 | 1 | $A+B+\overline{C}$ | $M_1$ |
| 0 | 1 | 0 | $A+\overline{B}+C$ | $M_2$ |
| 0 | 1 | 1 | $A+\overline{B}+\overline{C}$ | $M_3$ |
| 1 | 0 | 0 | $\overline{A}+B+C$ | $M_4$ |
| 1 | 0 | 1 | $\overline{A}+B+\overline{C}$ | $M_5$ |
| 1 | 1 | 0 | $\overline{A}+\overline{B}+C$ | $M_6$ |
| 1 | 1 | 1 | $\overline{A}+\overline{B}+\overline{C}$ | $M_7$ |

Important property of the maxterm is that it contains the value '0' f only one combination of K input variables, i.e., K-variable function of t $2^K$ maxterms, only one maxterm will possess the value 0, while all remaini $2^K-1$ maxterms will have the value 1 for an arbitrary input combinatio

***Q.43. State and prove De-Morgan's theorem.***

***[R.G.P.V., Dec. 2002(CS), June 2010, Nov. 201***

***Ans.*** **First Theorem** – It states that complement of two or more variabl when they are ORed together is same as those of complements of each inp when they are ANDed together. In other words, a NOR gate is equivalent to bubbles AND gate. For two input variables A and B

$$\overline{(A+B)} = \overline{A}.\overline{B}$$

The table 1.14 shows the truth table which shows the validity of this theorem –

**Table 1.14**

| *A* | *B* | $\overline{A}$ | $\overline{B}$ | $\overline{A+B}$ | $\overline{A}.\overline{B}$ |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 |

Fig. 1.19 shows the logic diagram for two input variables of the theorem.

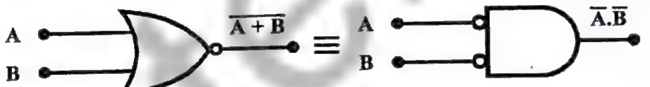


*Fig. 1.19*

The columns 5 and 6 of the truth table are identical. Hence we say that the two circuits are logically equivalent. Give the same inputs, the outputs are the same.

Similarly, for three input variables A, B, C

$$\overline{A+B+C} = \overline{A}.\overline{B}.\overline{C}$$

Fig. 1.20 shows the logic diagram for three input variables of the theorem.

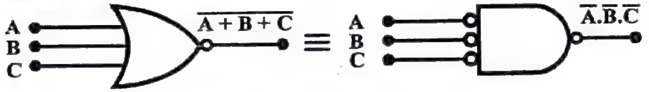


*Fig. 1.20*

The following truth table shows the validity of theorem for three input variables.

**Table 1.15**

| *A* | *B* | *C* | $\overline{A}$ | $\overline{B}$ | $\overline{C}$ | $\overline{A+B+C}$ | $\overline{A}.\overline{B}.\overline{C}$ |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |

**Second Theorem** – It states that complement of two or more variables when they are ANDed together is same as those of complement of each input when they are ORed together. In other words, a NAND gate is equivalent to a bubbled OR gate. For two input variables A, B –

$$\overline{A.B} = \overline{A}+\overline{B}$$

Fig. 1.21 shows the logic diagram for the two input variables of the theo

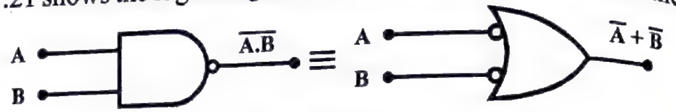


*Fig 1.21*

The following truth table shows the validity of this theorem –

**Table 1.16**

| $A$ | $B$ | $\overline{A}$ | $\overline{B}$ | $\overline{A.B}$ | $\overline{A}+\overline{B}$ |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 |

Similarly, for three-input variables A, B, C

$$\overline{(A.B.C)} = \overline{A}+\overline{B}+\overline{C}.$$

Fig. 1.22 shows the logic diagram for the three input variables of the theor

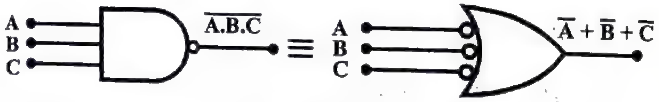


*Fig. 1.22*

The following truth table shows the validity of theorem for three in variables –

**Table 1.17**

| A | B | C | $\overline{A}$ | $\overline{B}$ | $\overline{C}$ | $\overline{A.B.C}$ | $\overline{A}+\overline{B}+\overline{C}$ |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |

The columns 7 and 8 of the truth table are identical. Hence we say that the tw circuits are logically equivalent. Given the same inputs, the outputs are the same

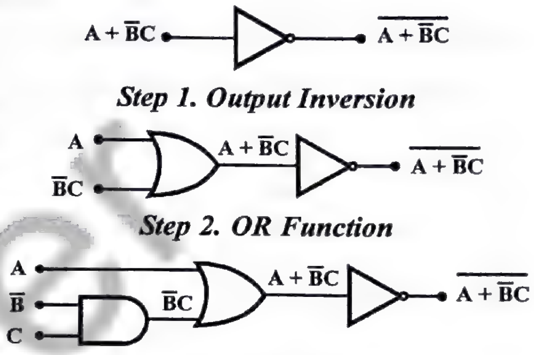
***Q.44. Define Boolean expressions and logic diagrams.***

***Ans.*** Boolean algebra is useless, unless it can be translated into hardwa in the form of AND gates, OR gates, and inverters. Similarly, Boolean algeb can be useful technique for analyzing existing circuits only if the hardw can be translated into a Boolean expression.

**Algebra to Logic –** The easiest way to convert an expression into a lo diagram is to start with the output and work toward the input. The expressi

$\overline{A+\overline{B}C}$ is to implemented in logic. As shown in fig. 1.23, start with the final expression. Since this is basically a noted function it must be the inverse of $A+\overline{B}C$. Note that the circle is on the output, indicating when $A+B\overline{C}$ is high, $\overline{A+\overline{B}C}$ is low. The circle should be drawn to indicate the location NOTed function.

Next consider the function on the input of the inverter, $A+\overline{B}C$. This is an OR of the term A and $\overline{B}C$ so draw an OR gate with two inputs. One input is term A, and other is term $\overline{B}C$.



***Step 1. Output Inversion***

***Step 2. OR Function***

***Step 3. AND Function***

***Fig. 1.23 Conversion of the Boolean to Logic Circuit***

The $\overline{B}C$ term is an AND function of $\overline{B}$ and C. Therefore, draw an AND gate with two inputs. Label the inputs B and C. The $\overline{B}$ term can be left as shown, or another inverter can be shown with a B input.

Many input to logic system are similar to the term $\overline{B}$ and enter the system as a 0 V signal when an event occurs. In this case, a B signal never exists in hardware.

Consider fig. 1.24, and assume the switch is ON when it is closed and off when open. Then signal is true when the switch output is a ground signal. If a + 5V signal were required to indicate the switch is ON, the $\overline{B}$ signal would have to feed an inverter whose output would be B. Thus, the NOT over the B indicates two things –

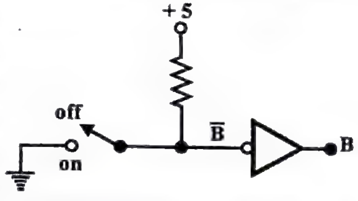


***Fig. 1.24 Switching to the Ground Terminal***

(i) The signal is true (high or 1) when a ground appears on the wire.

(ii) The signal must be inverted to obtain a high-level true signal.

***Q.45. Write down the important rules for reduction the Boolean expressions.***

***Ans.*** Rules for reduction the Boolean expression are given below –

(i) First select the term that has the fewest number of adjacencies.

(ii) Expand this expression to include the largest number of square possible in a $2^n$ configuration. Squares may be covered several times if necessary for this expansion.

(iii) Continue to select uncovered terms that have the fewest number of adjacencies.

(iv) All terms must be covered in the final expression.

(v) Finally, read the map. A two-square will eliminate one variab four-square two variable, an eight-square three variables, and a $2^n$ square, n varia

**Q.46. How the Boolean expressions are reduced ?**

**Ans.** As every element of hardware is represented by a logical open Boolean equation must be reduced to as simple a form as possible in ord reduce cost. The techniques, which are used in ordinary algebra, are also here for these reductions. The following procedure is the general approa

(i) Multiply all variables necessary to remove parantheses.

(ii) Look for identical terms. Only one of those terms be reta and all others dropped. For example,

$$AB+AB=AB$$

(iii) Look for a variable and its negation in the same term. This can be reduced. For example,

$$A\overline{A}C = 0.C$$
$$= 0$$ [∵ A.A

(iv) From the pairs of terms which are identical except for variable, the larger term can be dropped. For example,

$$ABCD+ABD = ABD\ (C+1)$$
$$= ABD.1 = ABD$$

If one variable is present in first term and its negation in the second this can be reduced. For example,

$$ABCD + A\overline{B}CD = (B+\overline{B})\,ACD$$
$$= 1.ACD = ACD$$

In the most of the cases, this procedure can be used.

## NUMERICAL PROBLEMS

**Prob.29. Prove the following Boolean identity –**

$$A + (B.C) = (A+B).(A+C)$$

(R.G.P.V., Nov. 2

**Sol.**
$$A + (B.C) = A.1 + BC$$ (∵ A.1
$$= A\,(1+B) + BC$$ (∵ 1 + B
$$= A.1 + AB + BC$$
$$= A\,(1+C) + AB + BC$$
$$= A + AC + AB + BC$$
$$= A.A + AC + AB + BC$$ (∵ A.A
$$= A\,(A+C) + B\,(A+C)$$
$$= \mathbf{(A+B)\,(A+C)}$$

Pr

**Prob.30. Minimize the following function using Boolean algebra –**

$$f_1 = AB\overline{C}D + \overline{BC + AD} + C(A + B\overline{DC})$$

$$f_2 = (BCD + A)\,\overline{B} + (\overline{\overline{AC} + \overline{BC}})\,D.$$

(R.G.P.V., Dec. 2012)

**Sol.** $f_1 = AB\overline{C}D + \overline{BC+AD} + C(A + B\overline{DC})$

$$= AB\overline{C}D + (\overline{BC})(\overline{AD}) + C(A + B(\overline{D}+\overline{C}))$$
$$= AB\overline{C}D + (\overline{B}+\overline{C})(\overline{A}+\overline{D}) + CA + BC\overline{D} + BC\overline{C}$$
$$= AB\overline{C}D + \overline{A}\,\overline{B} + \overline{B}\,\overline{D} + \overline{A}\,\overline{C} + \overline{C}\,\overline{D} + AC + BC\overline{D}$$ (∵ C.$\overline{C}$ = 0)
$$= \overline{A}\,\overline{B} + \overline{C}D + \overline{B}\,\overline{D} + \overline{A}\,\overline{C} + \overline{C}\,\overline{D} + AC + BC\overline{D}$$
$$= \overline{A}\,\overline{B} + \overline{B}\,\overline{D} + \overline{A}\,\overline{C} + \overline{C}(D+\overline{D}) + AC + BC\overline{D}$$
$$= \overline{A}\,\overline{B} + \overline{B}\,\overline{D} + \overline{A}\,\overline{C} + \overline{C}.1 + AC + BC\overline{D}$$
$$= \overline{A}\,\overline{B} + \overline{B}\,\overline{D} + \overline{C}(\overline{A}+1) + AC + BC\overline{D}$$
$$= \overline{A}\,\overline{B} + \overline{B}\,\overline{D} + \overline{C}.1 + AC + BC\overline{D}$$
$$= \overline{A}\,\overline{B} + \overline{B}\,\overline{D} + \overline{C} + A + BC\overline{D}$$
$$= A + \overline{A}\,\overline{B} + \overline{B}\,\overline{D} + \overline{C} + BC\overline{D}$$
$$= A + \overline{B} + \overline{B}\,\overline{D} + \overline{C} + B\overline{D}$$
$$= A + \overline{B} + \overline{C} + \overline{D}(B+\overline{B})$$
$$= \mathbf{A + \overline{B} + \overline{C} + \overline{D}}$$ **Ans.**

$$f_2 = (BCD + A)\overline{B} + \overline{(\overline{AC} + \overline{BC})}\,D$$
$$= \overline{B}.BCD + A\overline{B} + \overline{((\overline{A}+\overline{C}) + (\overline{B}+\overline{C}))}\,D$$
$$= 0 + A\overline{B} + \overline{(\overline{A}+\overline{C})}\;\overline{(\overline{B}+\overline{C})}\,D$$ [∵ B$\overline{B}$ = 0]
$$= A\overline{B} + \overline{\overline{A}}.\overline{\overline{C}}\,\overline{\overline{B}}\,\overline{\overline{C}}\,D$$
$$= A\overline{B} + ABCD = A(\overline{B} + BCD)$$
$$= A(\overline{B} + CD) = \mathbf{A\overline{B} + ACD}$$ **Ans.**

**Prob.31. Minimize using Boolean algebra –**

(i) $AB\overline{C} + BD + \overline{ABD} + AB\overline{C}D$

(ii) $A + B\overline{A} + ABC + \overline{AC} + \overline{A}B\overline{C}$.

(R.G.P.V., June 2012)

**Sol.** (i) $AB\overline{C} + BD + \overline{ABD} + AB\overline{C}D$

$$= AB\overline{C}(1+D) + BD + \overline{ABD}$$
$$= AB\overline{C} + BD + \overline{ABD}$$ (∵ 1 + D = 1)
$$= AB\overline{C} + BD + \overline{A} + \overline{B} + \overline{D}$$

$= \overline{A}+\overline{B}+\overline{D}+BD+AB\overline{C}$ $(\because \overline{A}+AB=\overline{A},$

$= \overline{A}+\overline{B}+\overline{D}+D+AB\overline{C}$

$= \overline{A}+\overline{B}+1+AB\overline{C}$

$= \overline{A}+1+AB\overline{C}$

$= 1+AB\overline{C} = \mathbf{1}$

*(ii)* $A+B\overline{A}+ABC+\overline{AC}+\overline{A}B\overline{C}$

$= A+B\overline{A}+ABC+\overline{AC}+\overline{A}B\overline{C}$ $(\because A+\overline{A}B=A,$

$= A+B+ABC+\overline{A}+\overline{C}+\overline{A}B\overline{C}$

$= A+\overline{A}+B+ABC+\overline{C}+\overline{A}B\overline{C}$

$= 1+B+ABC+\overline{C}+\overline{A}B\overline{C}$

$= 1+\overline{C}+ABC+\overline{A}B\overline{C}$

$= 1+ABC+\overline{A}B\overline{C}$

$= 1+\overline{A}B\overline{C} = \mathbf{1}$

***Prob.32. Show that (A + C) (A + D) (B + C) (B + D) = AB + CD,***

***(R.G.P.V., Dec. 2***

***Sol.*** Taking L.H.S.

$= (A+C)(A+D)(B+C)(B+D)$

$= (AA+AD+AC+CD)(BB+BD+BC+CD)$

$= AB+ABD+ABC+ACD+ABD+ABD+ABCD+ACD$
$+ABC+ABCD+ABC+ACD+BCD+BCD+BCD+$

$= AB+ABD+ABC+ACD+ABCD+BCD+CD$

$= AB(1+D)+ABC+ACD(1+B)+CD(B+1)$

$= AB+ABC+ACD+CD$

$= AB(1+C)+CD(A+1)$

$= AB+CD$

L.H.S. = R.H.S.

**Hence Pro**

***Prob.33. Simplify the following Boolean expression which repres the output of a logic decision***

$$F(A, B, C, D) = (AB+C+D)(\overline{C}+D)(\overline{C}+D+E)$$

***Sol.*** Given, ***(R.G.P.V., Nov. 2***

$F(A, B, C, D) = (AB+C+D)(\overline{C}+D)(\overline{C}+D+E)$

$= (AB\overline{C}+C\overline{C}+\overline{C}D+ABD+CD+D.D)(\overline{C}+D+E$

$= (AB\overline{C}+0+\overline{C}D+ABD+CD+D)(\overline{C}+D+E)$

$= AB\overline{C}+\overline{C}D+AB\overline{C}D+0+\overline{C}D+AB\overline{C}D+\overline{C}D+A$
$+CD+D+AB\overline{C}E+\overline{C}DE+ABDE+CDE+$

$= AB\overline{C}+AB\overline{C}D+\overline{C}D+ABD+CD+D+AB\overline{C}E$
$+\overline{C}DE+ABDE+CDE+DE$

$= AB\overline{C}(1+E)+AB\overline{C}D+D(C+\overline{C})+D+ABD(1+E)$
$+DE(C+\overline{C})+DE$

$= AB\overline{C}+AB\overline{C}D+D+ABD+DE$

$= AB\overline{C}(1+D)+ABD+D+DE$

$= AB\overline{C}+ABD+D(1+E)$

$= AB\overline{C}+ABD+D$ $(\because 1+E=1)$

$= \mathbf{AB(\overline{C}+D)+D}$ **Ans.**

***Prob.34. Simplify the following Boolean function to minimum numbers of literals –***

*(i)* $zx+zx'y$ *(ii)* $xy+xy'$ *(iii)* $y(wz'+wz)+xy$

***(R.G.P.V., June 2017)***

***Sol.*** (i) $zx+zx'y = z(x+x'y)$ $[A+A'B=A+B]$

$= \mathbf{z(x+y)}$ **Ans.**

(ii) $xy+xy' = x(y+y')$ $[A+A'=1]$

$= x.1$

$= \mathbf{x}$ **Ans.**

(iii) $y(wz'+wz)+xy = y[w(z+z')]+xy$

$= y[w.1]+xy$

$= yw+xy$

$= \mathbf{y(x+w)}$ **Ans.**

***Prob.35. Solve the following expressions –***

*(i)* $xy+\overline{xz}+x\overline{y}z(xy+z)$ *(ii)* $x.[y+z.(\overline{x.y+x.z})]$.

***(R.G.P.V., June 2011)***

***Sol.*** (i) $xy+\overline{xz}+x\overline{y}z(xy+z) = xy+\overline{xz}+x\overline{y}z.xy+x\overline{y}z.z$

$= xy+\overline{xz}+x\overline{y}z$ $[\because z.z=z \text{ and } \overline{y}.y=0]$

$= xy+\overline{x}+\overline{z}+x\overline{y}z$

$= \overline{x}+xy+\overline{z}+\overline{y}z$ $[\because x+\overline{x}y=x+y]$

$= \overline{x}+xy+\overline{z}+z\overline{y}$

$= \overline{x}+y+\overline{z}+\overline{y}$

$= \overline{x}+\overline{z}+1$ $[y+\overline{y}=1]$

$= \overline{x}+1$

$= \mathbf{1}$ **Ans.**

(ii) $x.[y+z.(\overline{x.y+x.z})] = x[y+z(\overline{x.y}).(\overline{x.z})]$

$$= [xy+xz.(\overline{x.y}).(\overline{x.z})]$$
$$= xy$$
$$= \mathbf{xy}$$

***Prob.36. Obtain the truth table of the function –***

$F = xy + xy' + y'z$ ***(R.G.P.V., Dec.***

*Sol.* The truth table of the given function is shown in table 1.18.

**Table 1.18**

| *Input Variables* | | | | | | | *Output* |
|---|---|---|---|---|---|---|---|
| *x* | *y* | *z* | *y'* | *xy* | *xy'* | *y'z* | *F = xy + xy' + y'z* |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |

***Prob.37. Write the minterm of ACD + AB and implement it.***

***(R.G.P.V., June 2***

*Sol.* $y = ACD + AB$

$$= ACD(B+\overline{B}) + AB(C+\overline{C})(D+\overline{D})$$
$$= ABCD + A\overline{B}CD + (ABC + AB\overline{C})(D+\overline{D})$$
$$= ABCD + A\overline{B}CD + ABCD + ABC\overline{D} + AB\overline{C}D + AB\overline{C}$$
$$= ABCD + A\overline{B}CD + ABC\overline{D} + AB\overline{C}D + AB\overline{C}\,\overline{D}$$

The implementation of above equation is shown in fig. 1.25.

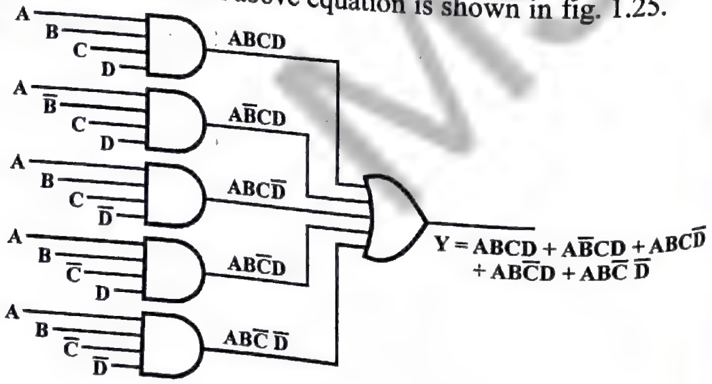


*Fig. 1.25*

***Prob.38. Express the following in sum of minterms and product of maxterm form –***

*(i)* $f(A, B, C) = (\overline{A}+B)(\overline{B}C)$

*(ii)* $f(x, y, z) = 1.$ ***(R.G.P.V., June 2006)***

***Sol.*** **(i) (a) Sum of Minterm –**

$$f(A, B, C) = (\overline{A}+B)(\overline{B}C)$$
$$= \overline{A}\,\overline{B}C + B\overline{B}C \qquad \begin{Bmatrix} \overline{B}B = 0 \\ 0.C = 0 \end{Bmatrix}$$
$$= \overline{A}\,\overline{B}C$$
$$f(A, B, C) = m_1$$
$$f(A, B, C) = \Sigma m(1)$$

**(b) Product of Maxterm –**

The expression is a three-variable function. The variable C is missing in the first term, so add $C\overline{C}$ to it. Therefore,

$$(\overline{A}+B) = (\overline{A}+B+C\overline{C}) = (\overline{A}+B+\overline{C})(\overline{A}+B+C)$$

Similarly

$$\overline{B} = \overline{B}+A\overline{A}+C\overline{C} = (A+\overline{B}+C\overline{C})(\overline{A}+\overline{B}+C\overline{C})$$
$$= (A+\overline{B}+\overline{C})(A+\overline{B}+C)(\overline{A}+\overline{B}+\overline{C})(\overline{A}+\overline{B}+C)$$
$$C = C+A\overline{A}+B\overline{B} = (\overline{A}+C+B\overline{B})(A+C+B\overline{B})$$
$$= (\overline{A}+\overline{B}+C)(\overline{A}+B+C)(A+\overline{B}+C)(A+B+C)$$

Now,

$$(\overline{A}+B)\overline{B}C = (\overline{A}+\overline{B}+\overline{C})(\overline{A}+\overline{B}+C)(\overline{A}+B+\overline{C})(\overline{A}+B+C)$$
$$(A+\overline{B}+\overline{C})(A+\overline{B}+C)(A+B+C)$$
$$f(A, B, C) = M_7, M_6, M_5, M_4, M_3, M_2, M_0$$
$$= \Pi\, M\ (0, 2, 3, 4, 5, 6, 7)$$

*(ii)* $f(x, y, z) = 1$

**(a) Minterm** – The expression is a three variable function. Function value is equal to 1 it means that all ($2^n = 8$) squares are adjacent and contain 1. Therefore minterm will be,

$$f(x, y, z) = \overline{x}\overline{y}\overline{z}+\overline{x}\overline{y}z+\overline{x}y\overline{z}+\overline{x}yz+x\overline{y}\overline{z}+x\overline{y}z+xy\overline{z}+xyz$$
$$= m_0 + m_1 + m_2 + m_3 + m_4 + m_5 + m_6 + m_7$$
$$= \sum m\ (0, 1, 2, 3, 4, 5, 6, 7)$$

**(b) Maxterm** – Taking the complement of minterms, we obtain the maxterm.

$$\text{Minterm} = \overline{x}\overline{y}\overline{z}+\overline{x}\overline{y}z+\overline{x}y\overline{z}+\overline{x}yz+x\overline{y}\overline{z}+x\overline{y}z+xy\overline{z}+xyz$$
$$\text{Maxterm} = (x+y+z)(x+y+\overline{z})(x+\overline{y}+z)(x+\overline{y}+\overline{z})$$
$$(\overline{x}+y+z)(\overline{x}+y+\overline{z})(\overline{x}+\overline{y}+z)(\overline{x}+\overline{y}+\overline{z})$$
$$= M_0\, M_1\, M_2\, M_3\, M_4\, M_5\, M_6\, M_7$$
$$= \Pi M\ (0, 1, 2, 3, 4, 5, 6, 7)$$

***Prob.39. Prove the following using De-Morgan's theorem –***

$$(A + B)\,(C + D) = \overline{(\overline{A+B}) + (\overline{C+D})}$$

***and hence prove that an OR-AND configuration is equivalent to NOR configuration. Realize the logic equation given above using –***

***(i) OR and AND gates (ii) Only NOR gates.***

***(R.G.P.V., June 2008,***

***Sol.*** $(A + B)\,(C + D) = (A + B).(C + D)$

$$= (\overline{\overline{A+B}}).(\overline{\overline{C+D}}) = \overline{(\overline{A+B}) + (\overline{C+D})}$$

The left hand side is realizable by using two 2-input OR gates follow a 2-input AND gate, while the right hand side is realizable by two 2-input gates followed by another 2-input NOR gate. Hence, an OR-AND configu is equivalent to a NOR-NOR configuration.

The realization equation when OR and AND gates.

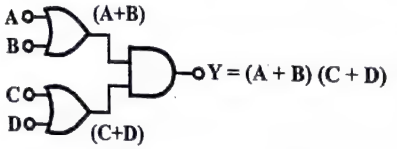


*Fig. 1.26*

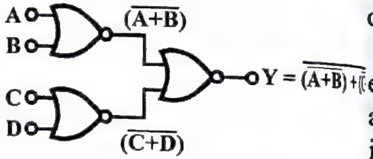


*Fig. 1.27*

The realization equation when only NOR gates.

## KARNAUGH MAP METHODS, SOP-POS SIMPLIFICATIO NAND-NOR IMPLEMENTATION

***Q.47. Write short note on Karnaugh's map methods.***

***(R.G.P.V., Nov./Dec. 2***

***Or***

***What is Karnaugh Map ?***

***(R.G.P.V., Dec. 2***

***Ans.*** The Karnaugh's map technique provides a systematic method simplifying and manipulating switching expressions.

In this technique, the information contained in a truth table or availab the POS or SOP form is represented on the Karnaugh map or K-map.

The K-map actually modified form of a truth table. Here, the combinat are conveniently arranged to aid the simplification process by applying the Ax + Ax' = A. In an n-variable K-map, there are $2^n$ cells.

Each cell corresponds to one combination of n-variables. Therefore each row of the truth table, i.e., for each minterm and for each maxt there is one specific cell in the K-map.

A two variable K-map has $2^2 = 4$ squares. These squares are called cells. Each square on the K-map represents a unique minterm. The minterm designations of the squares are shown in fig. 1.28. A 1 placed in any square indicates that the corresponding minterm is included in the output expression, and a 0 or no entry in any square indicates that the corresponding minterm does not appear in the expression for output.

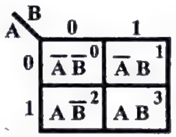


***Fig. 1.28 Two-variable K-map***

***Q.48. Describe don't care condition with the aid of example.***

***Ans.*** Some logic circuits can be designed, so, there are certain input conditions for which there are no specified output levels, usually because these input conditions will never occur. In other words, there will be certain combinations of input levels where we "don't care" whether the output is high or low.

The K-maps are simplified using either 1's or 0's. Therefore, we make the entries in the K-map for either 1's or 0's. The cells, which do not contain 1 are assumed to contain 0 and vice-versa. This is not always true since there are cases in which certain combinations of input variables do not occur. Also, for some functions the outputs corresponding to certain combinations of input variables do not matter. In such situations the designer has a flexibility and it is left to him whether to assume a 0 or a 1 as output for each of these combinations. This condition is known as don't care condition and can be represented on the K-map as a cross mark (×) in the corresponding cell. The cross mark (×) in a cell may be assumed to be a 1 or a 0 depending upon which one leads to a simpler expression. The function can be specified in one of the following manners –

(i) For example, consider the case of minterms with don't care conditions as follows –

$$f\,(A, B, C, D) = \Sigma m\,(1, 3, 7, 11, 15) + d\,(0, 2, 5) \qquad \text{...(i)}$$

The K-map of equation (i) is shown in fig. 1.29.

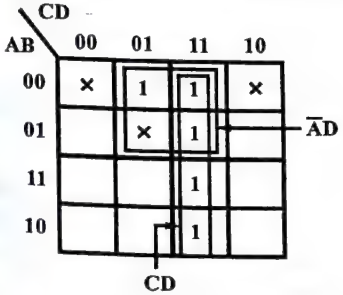


***Fig. 1.29 K-map and Minimized Expression of Equation (i)***

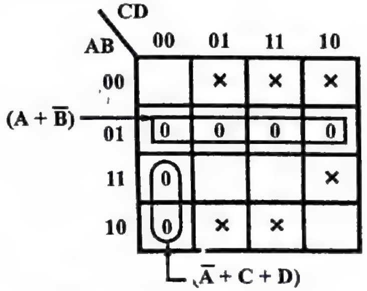


***Fig. 1.30 K-map and Minimized Expression of Equation (ii)***

(ii) The simplified expression of the above K-map is as follo maxterms with don't care conditions as follows –

f (A, B, C, D) = Π M (4, 5, 6, 7, 8, 12) + d (1, 2, 3, 9, 11, 14)

The K-map of equation (ii) is shown in fig. 1.30.

***Q.49. What do you understand by prime implicant ? Explain wi aid of an example.***

***Or***

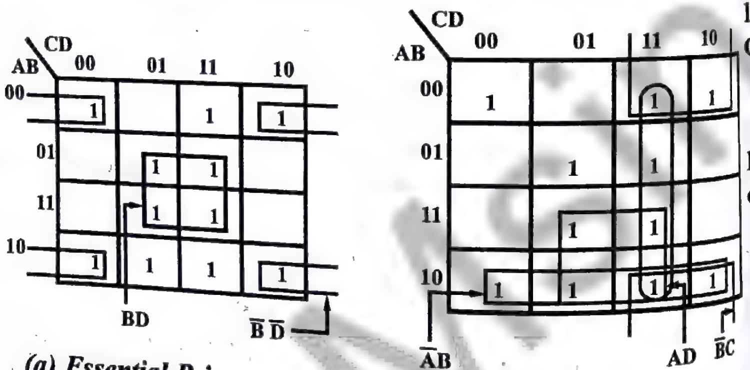
***Write short note on the concept of prime implicant.***

***(R.G.P.V., June***

***Ans.*** A prime implicant is a product term obtained by combinin maximum possible number of adjacent squares in the map. When a minte a square is covered by only one prime implicant, then that implicant is s be extremely important. The prime implicants of a function can be obt from the map by combining all possible maximum numbers of squar single 1 on a K-map represents a prime implicant if it is not adjacent t other 1's. Two adjacent 1's form a prime implicant provided that they ar within a group of four adjacent squares. Four adjacent 1's form a prime imp if they are not within a group of eight adjacent squares and so on.

For example, consider the following four-variable Boolean functio

F (A, B, C, D) = Σ (0, 2, 3, 5, 7, 8, 9, 10, 11, 13, 15)



***(a) Essential Prime Implicants B.D and $\overline{B}.\overline{D}$***

***(b) Prime Implicants CD, $\overline{B}C$, AD and $A\overline{B}$***

***Fig. 1.31***

The minterms of the function are marked with 1's in the K-map fig. 1.31. Two essential prime implicants are shown in fig. 1.31 (a). One is essential because there is one way to include minterms $m_0$ within adjacent squares. These four squares define the term $\overline{B}.\overline{D}$.

In a similar manner, there is only one way that minterms $m_5$, $m_7$ and $m_{15}$ forms a square and gives the term B.D. The two essential p

implicants cover eight minterms. Fig. 1.31 (b) shows all possible ways that minterms $m_3$, $m_9$ and $m_{11}$ can be covered with prime implicants. The minterm $m_3$ can be covered with either prime implicant CD or $\overline{B}C$. Minterm $m_9$ can be covered with either AD or $A\overline{B}$. Minterm $m_{11}$ is covered with any one of the four prime implicants.

The simplified expression is obtained from the logical sum of the two essential prime implicants and any two prime implicants that cover minterms $m_3$, $m_9$ and $m_{11}$. There are four possible ways that the function can be expressed with four product terms of two literals each, which is given as –

$$
\begin{aligned}
F &= BD+\overline{B}\,\overline{D}+CD+AD \\
&= BD+\overline{B}\,\overline{D}+CD+A\overline{B} \\
&= BD+\overline{B}\,\overline{D}+\overline{B}C+AD \\
&= BD+\overline{B}\,\overline{D}+\overline{B}C+A\overline{B}
\end{aligned}
$$

The identification of the prime implicants in the K-map helps in determining the alternatives that are available for obtaining a simplified expression.

***Q.50. Define the followings –***

***(i) SOP (ii) POS.***

***Ans. (i) Sum of Products (SOP)*** – The logical sum of two or more logical product terms, is known as a sum of products (SOP). It is basically an OR operation of AND operated variables such as –

$$Y = AB + B\overline{C} + AC$$

***(ii) Product of Sums (POS)*** – A product of sums expression is a logical product of two or more logical sum terms. It is basically an AND operation of OR operated variables such as –

$$Y = (A+B)(B+\overline{C})(A+C)$$

***Q.51. How do you convert an SOP form to a POS form and vice versa?***

***Ans. (i) Canonical (Sum of Products) SOP Form*** – In the given expression, the variables B and C are missing in the first term. The variable A is missing in the second term. Hence first term has to be multiplied by $(B+\overline{B})$ and $(C+\overline{C})$, the second term by $(A+\overline{A})$.

$$
\begin{aligned}
Y &= A+\overline{B}C \\
&= A(B+\overline{B})(C+\overline{C})+\overline{B}C(A+\overline{A}) \\
&= A(BC+B\overline{C}+\overline{B}C+\overline{B}\,\overline{C})+A\overline{B}C+\overline{A}\,\overline{B}C \\
&= ABC+AB\overline{C}+A\overline{B}C+A\overline{B}\,\overline{C}+A\overline{B}C+\overline{A}\,\overline{B}C \\
&= ABC+AB\overline{C}+A\overline{B}C+A\overline{B}\,\overline{C}+\overline{A}\,\overline{B}C
\end{aligned}
$$

*(ii)* ***Canonical Product of Sum (POS) Form –***

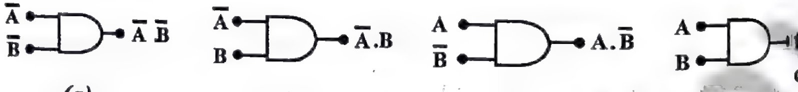
$$Y = A + \overline{B}C$$
$$= (A + \overline{B})(A + C)$$
$$[\because (A + BC) = (A + B)(A_{+}$$

In this expression, the variable C is missing in the first term. The vari B is missing in the second term. Hence, the first term has to be added $C\overline{C}$, the second term with $B\overline{B}$.

$$Y = (A + \overline{B} + C\overline{C})\,(A + B\overline{B} + C)$$
$$= (A + \overline{B} + C)(A + \overline{B} + \overline{C})(A + B + C)(A + \overline{B} + C)$$
$$= (A + \overline{B} + C)(A + \overline{B} + \overline{C})(A + B + C)$$

***Q.52. Write the sum-of products (SOP) method with the help of t table and logic circuit.***

***Ans.*** There are four possible way to AND two input signals which a complemented and uncomplemented form, shown in fig. 1.32. These out are known as fundamental products. Each fundamental product next to input conditions producing a high output is listed in table 1.19.



*(a)* *(b)* *(c)* *(d)*

***Fig. 1.32 ANDing Two Variables and Their Complements***

**Table 1.19 Fundamental Products Two Inputs**

| A | B | *Fundamental Product* |
|---|---|---|
| 0 | 0 | $\overline{A}\ \overline{B}$ |
| 0 | 1 | $\overline{A}\ B$ |
| 1 | 0 | $A\ \overline{B}$ |
| 1 | 1 | $A\ B$ |

The fundamental products idea applies to three or more input variables. Let us take three input variables; A, B, C and their complements. To AND these three input variables and their complements, there are eight possible ways. It results in fundamental products of

$$ABC, AB\overline{C}, A\overline{B}C, A\overline{B}\,\overline{C}, \overline{A}BC, \overline{A}B\overline{C}, \overline{A}\,\overline{B}C, \overline{A}\,\overline{B}\,\overline{C}$$

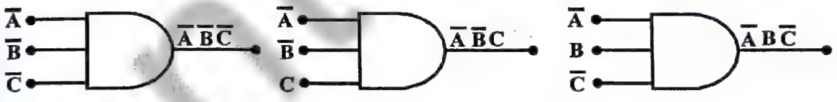
Fig. 1.33 (a) shows the last fundamental product, fig. 1.33 (b) the sec last and Fig. 1.33 (c) the third from the last.

Table 1.20 summarizes the fundamental products, listing each one to the input condition which results in a high output. For example, if A B = 0 and C = 0, the fundamental product obtained in an output form of

$$y = A\overline{B}\,\overline{C} = 1\,0\,0 = 1$$

**Table 1.20 Fundamental Products for Three Inputs**

| *A* | *B* | *C* | *Fundamental Product* |
|---|---|---|---|
| 0 | 0 | 0 | $\overline{A}\ \overline{B}\ \overline{C}$ |
| 0 | 0 | 1 | $\overline{A}\ \overline{B}\ C$ |
| 0 | 1 | 0 | $\overline{A}\ B\ \overline{C}$ |
| 0 | 1 | 1 | $\overline{A}\ B\ C$ |
| 1 | 0 | 0 | $A\ \overline{B}\ \overline{C}$ |
| 1 | 0 | 1 | $A\ \overline{B}\ C$ |
| 1 | 1 | 0 | $A\ B\ \overline{C}$ |
| 1 | 1 | 1 | $A\ B\ C$ |



*(a)* *(b)* *(c)*

***Fig. 1.33 Examples of ANDing Three Variables and Their Complements***

**Sum of Products Equation** – To get the sum of products solution, a truth table is given like table 1.21. Fundamental product is written for each output 1 in the truth table. For example, the first output 1 appears for an input of A = 0, B = 1 and C = 1. Then the fundamental product will be $\overline{A}BC$. For A = 1, B = 0 and C = 1, the next output 1 appears. $A\overline{B}C$ is the fundamental product for this.

**Logic Circuit** – After getting sum-of-products equation, logic circuit can be drawn by an AND-OR network, or a NAND-NAND network. The AND-OR circuit is the one solution of the problem. The AND-OR circuit of fig. 1.34 has the truth table given by table 1.21.

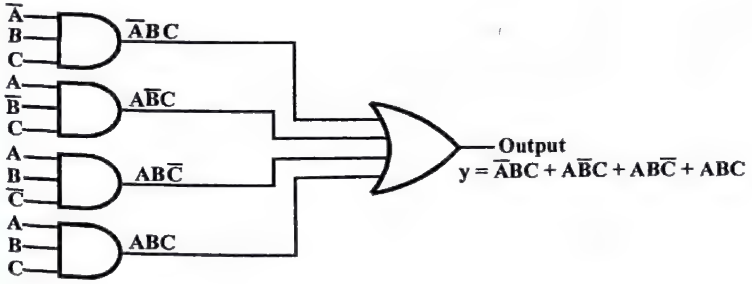


***Fig. 1.34 Design of AND-OR Logic Gates***

Table 1.21 Design Truth Table

| *A* | *B* | *C* | *Output* |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | $1 \Rightarrow \overline{A}BC$ |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | $1 \Rightarrow A\overline{B}C$ |
| 1 | 1 | 0 | $1 \Rightarrow AB\overline{C}$ |
| 1 | 1 | 1 | $1 \Rightarrow ABC$ |

***Q.53. Explain the product-of-sum (POS) method with the help of table and logic circuit.***

***Ans.*** In the product-of-sums method, a truth table is given an fundamental sums are required for a logic design. By ANDing these sum product-of-sums equation is obtained corresponding to the truth table. are few differences between sum-of-product and product-of-sum approa In sum-of-products method, the fundamental product produces an outp for the corresponding input condition. On the other hand in the produ sums method, the fundamental sum produces an output 0 for the correspo input condition.

**Converting a Truth Table to an Equation** – Suppose, a truth table is given in table 1.22. To get the product-of-sums equation, we locate each output 0 in the truth table and write down its fundamental sum. The first output 0 appears for A = 0, B = 0 and C = 0 in table 1.22. The fundamental sum for these inputs is A + B + C and this produces an output zero for the corresponding input condition,

$$\text{Output} = A + B + C = 0 + 0 + 0 = 0$$

Table 1.22

| *A* | *B* | *C* | *Y* |
|---|---|---|---|
| 0 | 0 | 0 | $0 \Rightarrow A + B +$ |
| 0 | 0 | 1 | $0 \Rightarrow A + B +$ |
| 0 | 1 | 0 | $0 \Rightarrow A + \overline{B} +$ |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | $0 \Rightarrow \overline{A} + B +$ |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

The second output 0 appears for the input condition of A = 0, B = C = 1. A +B + $\overline{C}$ is the fundamental sum for this. To get a logical sum of the given input conditions, C is complemented

$$\text{Output} = A + B + \overline{C} = 0 + 0 + \overline{1}$$
$$= 0 + 0 + 0 = 0$$

The third output 0 occurs for A = 0, B = 1 and C = 0. Its fundam sum is A + $\overline{B}$ + C

$$\text{Output} = A + \overline{B} + C = 0 + \overline{1} + 0 = 0 + 0 + 0 = 0$$

Similarly, the fourth output 0 appears for the input condition of A = 1, B = 0, C = 0. Hence its fundamental same is $\overline{A}$ + B + C

$$\text{Output} = \overline{A} + B + C = \overline{1} + 0 + 0 = 0 + 0 + 0 = 0$$

Table 1.22 shows all the fundamental sums required to implement the truth table. Notice that when input variable is 1, then corresponding variable is complemented and when input variable is 0, the corresponding variable is uncomplemented. By ANDing the fundamental sums, product-of-sums equation is obtained

$$Y = (A + B + C)\,(A + B + \overline{C})\,(A + \overline{B} + C)\,(\overline{A} + B + C) \qquad \text{...(i)}$$

This is the product-of-sums equation for table 1.22.

**Logic Circuit** – After getting a product-of-sums equation, the logic circuit can be got by drawing an OR-AND network, or a NOR-NOR network. In equation (i) each sum shows the output of a 3-input OR gate and the logical Y is the output of a 4-input AND gate. Hence, a circuit can be drawn as shown in fig. 1.35.

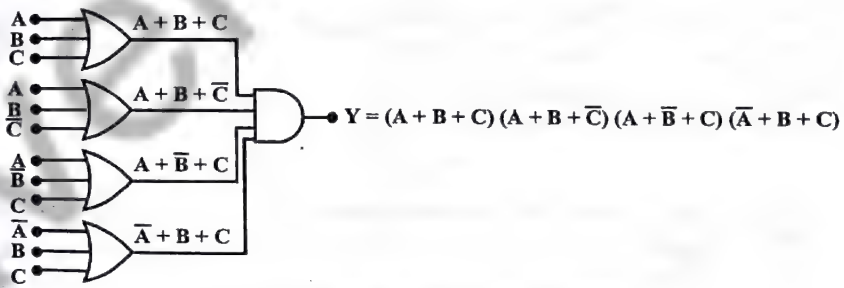


*Fig. 1.35 Product-of-sums Circuit*

***Q.54. How can be converted to NOR-NOR and NAND-NAND gate networks ?***

***Ans.*** According to De-Morgan's theorem,

$$\overline{AB} = \overline{A} + \overline{B}$$

and

$$\overline{A + B} = \overline{A}\,\overline{B}$$

The above expressions may be written as

$$AB = \overline{\overline{A} + \overline{B}} \qquad \text{...(i)}$$

and

$$A + B = \overline{\overline{A}\,\overline{B}} \qquad \text{...(ii)}$$

Hence, an AND gate is equivalent to a NOR gate with bubbles at its inputs and an OR gate is equivalent to a NAND gate with bubbles at its inputs. Also, a NOR gate is equivalent to an AND gate with bubbles at its inputs and a NAND gate is equivalent to an OR gate with bubbles at its inputs, as discussed in Q.28. The above equations (i) and (ii) are implemented in fig. 1.36 (a) and (b).

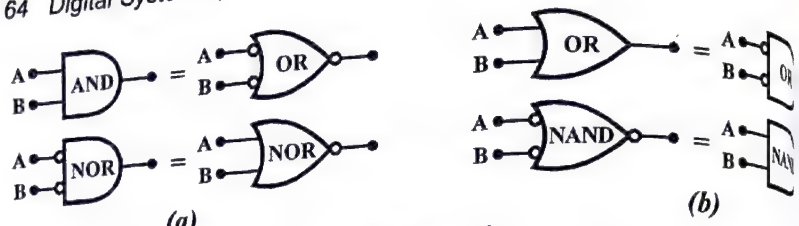


Fig. 1.36

**Q.55. Write short note on AND-OR conversion to NOR-NOR gate netv**

*Ans.* Fig. 1.37 (a) shows a two level AND-OR gate network. A two- AND-OR gate network is converted into NOR-NOR gate network by rep the AND gate with NOR gates and the OR gate with a NOR gate. But it logically equivalent.

At the inputs and output, it is corrected by using inverters as sho fig. 1.37 (b). Fig. 1.37 (c) shows the modified circuit of fig. 1.37 (b).

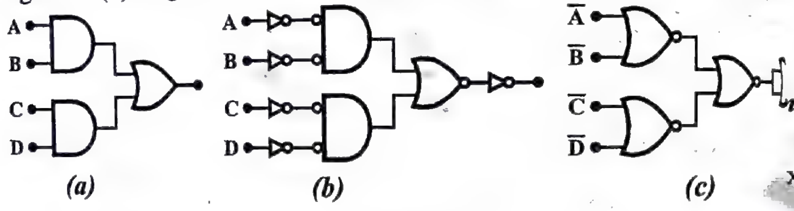


Fig. 1.37

**Q.56. Explain in brief OR-AND conversion to NAND-NAND gate netw**

*Ans.* A two level OR-AND gate network is shown in fig. 1.38 (a). I converted into NAND-NAND gate network by replacing the AND gate w NAND gate and OR gate with NAND gates. But this is not logically equiv at the inputs and output, it is corrected by using inverters as shown i 1.38 (b). Fig. 1.38 (c) shows the modified circuit of fig. 1.38 (b).

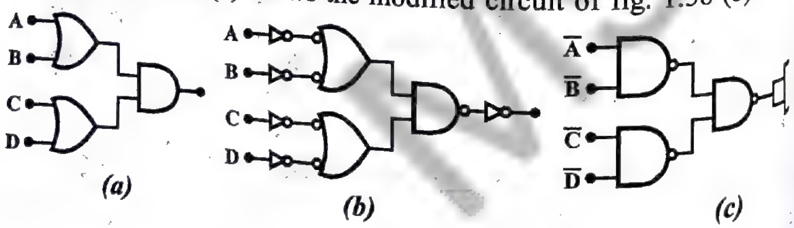


Fig. 1.38

**Q.57. How can be converted to AND-OR into NAND-NAND gate netw**

*Ans.* A multilevel AND-OR gate network is simply converted into network with NAND-NAND gate network. This is logically equivalen gate with the uncomplemented switching variable and a bubble at the in that gate to which the complemented switching variable can be conn

henever a gate has a complemented switching ariable at its input. Two bubbles are shown t both ends of a line connecting two gates ince their is no logical effect. According to e above conditions, the circuit is depicted in ig. 1.39 (a). Fig. 1.39 (b) shows the modified ircuit of fig. 1.39 (a).

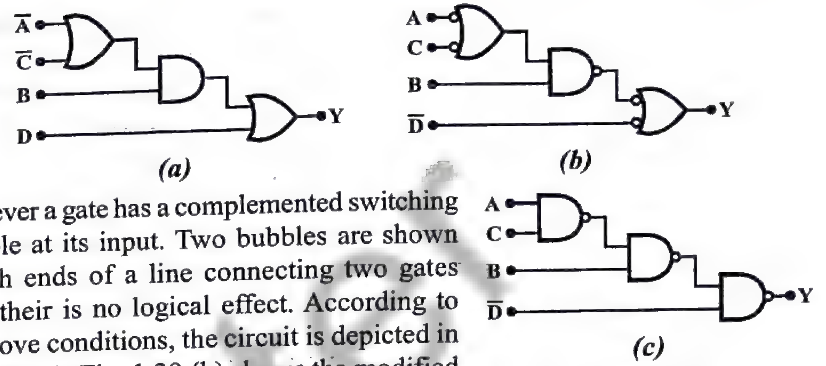


Fig. 1.39

## NUMERICAL PROBLEMS

**Prob.40. Prove sum of equation $Y = ABCD + ABC\overline{D}$ using Karnaugh maps.** *(R.G.P.V., Dec. 2013)*

*Sol.* The given expression can be xpressed as –

$$f(A, B, C, D) = \Sigma m\ (14, 15) \quad ...(i)$$

The K-map for the equation (i) is illustrated in fig. 1.40.

| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | | | | |
| 01 | | | | |
| 11 | | | 1 | 1 |
| 10 | | | | |

Fig. 1.40 K-map for f(A, B, C, D) = Σm (14, 15)

The minimized expression obtained from the K-map is given as under –

$$Y = ABC$$

**Prob.41. Simplify the following Boolean function with K-map –**
**$F(w, x, y, z) = \Sigma(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)$.**
*(R.G.P.V., June 2010, 2011, 2014)*

***Or***

**Simplify the Boolean function using K-map.**
**$F(A, B, C, D) = \Sigma(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)$**
*(R.G.P.V., Dec. 2017)*

*Sol.* Function F has four variables so a four ariable map must be used.

K-map of given function F is shown in fig. 1.41.

| wx \ yz | y'z' | y'z | yz | yz' | |
|---|---|---|---|---|---|
| w'x' | $^{0}$ 1 | $^{1}$ 1 | $^{3}$ | $^{2}$ 1 | w'z' |
| w'x | $^{4}$ 1 | $^{5}$ 1 | $^{7}$ | $^{6}$ 1 | |
| wx | $^{12}$ 1 | $^{13}$ 1 | $^{15}$ | $^{14}$ 1 | xz' |
| wx' | $^{8}$ 1 | $^{9}$ 1 | $^{11}$ | $^{10}$ | |
| y' | | | | | |

Fig. 1.41

The minimized expression is given below –

$$F = y' + w'z' + xz' \qquad \textbf{Ans.}$$

**Prob.42. *Minimize the given function using K-map and com minimize function into POS form –***

***F(A, B, C, D) = (1, 3, 5, 7, 9, 10, 12, 13)*** ***(R.G.P.V., No***

***Sol.*** The given expression is

F(A, B, C, D) = (1, 3, 5, 7, 9, 10, 12, 13)

The K-map for the above expression is shown in fig. 1.42.

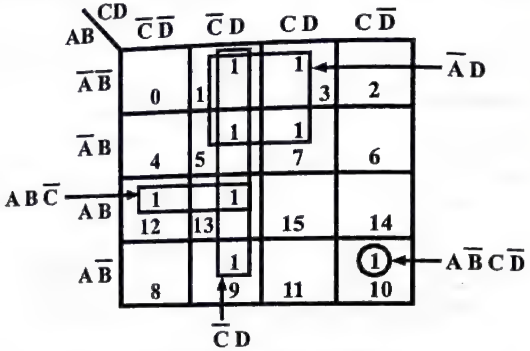


***Fig. 1.42 K-map for SOP Form***

The simplified expression in SOP form is given as

$$F(A, B, C, D) = \bar{A}D + \bar{C}D + AB\bar{C} + A\bar{B}C\bar{D}$$

The simplified expression in POS form is given as

$$F(A, B, C, D) = (B + C + D)(A + D)(\bar{A} + \bar{B} + \bar{C})(\bar{A} + $$

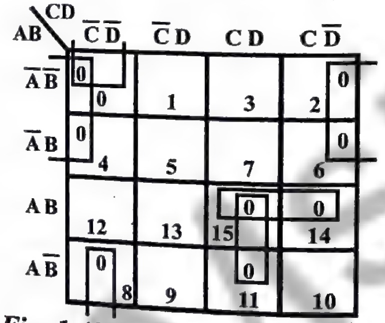


***Fig. 1.43 K-map for POS Form***

**Prob.43. *Simplify the Boolean function –***

***F(w, x, y, z) = Σm(1, 3, 7, 11, 15)***

***+ Σd(0, 2, 5).***

***(R.G.P.V., June 2014)***

***Sol.*** The K-map for the given function with don't care condition is shown in fig. 1.44.

F = (w, x, y, z)

= Σm(1, 3, 7, 11, 15) + Σd(0, 2, 5)

The simplified expression is

$$F = \bar{w}\bar{x} + yz$$

**Ans.**

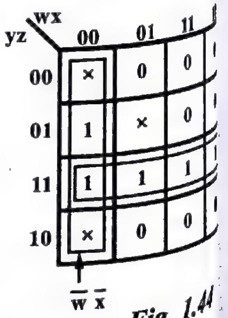


*Fig. 1.44*

**Prob.44. *Simplify the Boolean function with don't care conditions and implement it with NAND gates –***

***F(w, x, y, z) = Σ (1, 3, 7, 11, 15)***

***d (w, x, y, z) = Σ (0, 2, 5)***

***(R.G.P.V., May 2018)***

***Sol.*** Refer the sol. of Prob.43.

The simplified expression is given as

$$F = \bar{w}\bar{x} + yz$$

The implementation of above minimized function using NAND gates is shown in fig. 1.45.

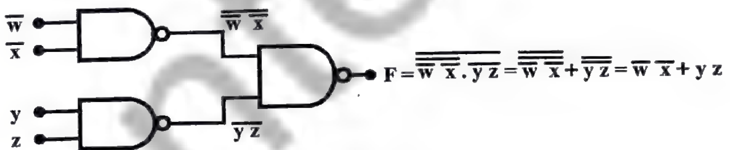

$\bar{w}$
$\bar{x}$
y
z
$\overline{\bar{w}\bar{x}}$
$\overline{yz}$
$F = \overline{\overline{\bar{w}\bar{x}}.\overline{yz}} = \overline{\overline{\bar{w}\bar{x}}} + \overline{\overline{yz}} = \bar{w}\bar{x} + yz$

*Fig. 1.45*

**Prob.45. *Simplify the Boolean function***

***F = B'C'D' + BCD' + ABCD'***

***and the don't care condition***

***d = B'CD' + A'BC'D*** ***(R.G.P.V., June 2017)***

***Sol.*** Given, F = B'C'D' + BCD' + ABCD'

The given function is of four variables. Hence,

F = (A + A') (B'C'D') + (A + A') (BCD') + ABCD'

= AB'C'D' + A'B'C'D' + ABCD' + A'BCD' + ABCD'

= AB'C'D' + A'B'C'D' + A'BCD' + ABCD'

F(A, B, C, D) = Σm(0, 6, 8, 14) + Σd(2, 5, 10)

The K-map for above function is shown in fig. 1.46.

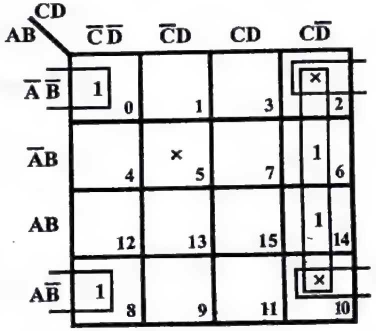


*Fig. 1.46*

The minimized expression is given below –

$$F(A, B, C, D) = \bar{B}\bar{D} + C\bar{D}$$

**Ans.**

***Prob.46. Minimize the following function using Karnaugh map me***

***$F(a, b, c, d) = \Sigma m\ (1, 3, 7, 10, 12, 14) + \Sigma d\ (0, 9)$***

**(R.G.P.V., Dec**

***Sol.*** The K-map for the above expression is shown in fig. 1.47.

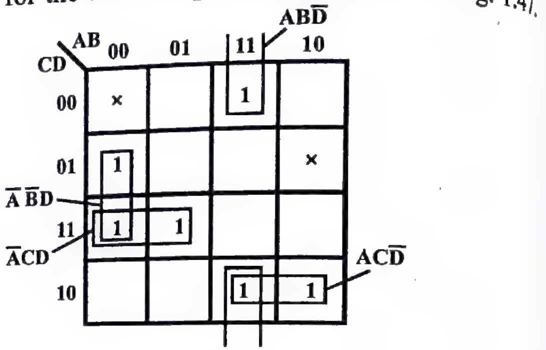


*Fig. 1.47*

$$Y = \bar{A}\,\bar{B}\,D + \bar{A}CD + AC\bar{D} + AB\bar{D}$$

***Prob.47. Minimize the given Boolean function using K-ma implement the simplified function using only NAND gates.***

***$F(A, B, C, D) = \Sigma m(0, 1, 2, 9, 11, 15) + d(8, 10, 14)$.***

**(R.G.P.V., Dec.**

***Sol.*** Function F has four variables so a four variable map must b

$F(A, B, C, D) = \Sigma m(0, 1, 2, 9, 11, 15) + d(8, 10, 14)$

The K-map for the given function with don't care condition is shown in fig. 1.48.

The simplified expression is,

$$F = AC + \bar{B}\,\bar{C} + \bar{B}\,\bar{D}$$

Implementation of above minimized function using only NAND gate is shown in fig. 1.49.

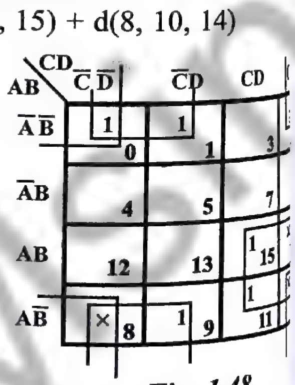
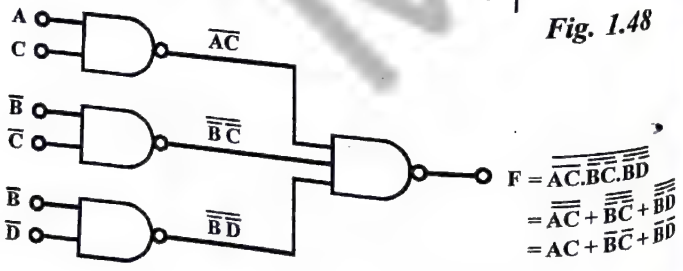


*Fig. 1.48*



A
C
$\overline{AC}$
$\bar{B}$
$\bar{C}$
$\overline{\bar{B}\,\bar{C}}$
$\bar{B}$
$\bar{D}$
$\overline{\bar{B}\,\bar{D}}$
$F = \overline{\overline{AC}.\overline{\bar{B}\bar{C}}.\overline{\bar{B}\bar{D}}}$
$= \overline{\overline{AC}} + \overline{\overline{\bar{B}\bar{C}}} + \overline{\overline{\bar{B}\bar{D}}}$
$= AC + \bar{B}\bar{C} + \bar{B}\bar{D}$

*Fig. 1.49*

***Prob.48. Minimize the following switching functions using the Karnaugh map. List all prime implicants and essential prime implicants –***

***$F(x_1, x_2, x_3, x_4) = \Sigma\ (0, 1, 2, 3, 6, 7, 9, 13, 14, 15)$.***

**(R.G.P.V., Dec. 2010)**

***Sol.*** The given function is

$F(x_1, x_2, x_3, x_4) = \Sigma(0, 1, 2, 3, 6, 7, 9, 13, 14, 15)$

K-map for given function is shown in fig. 1.50.

The minimal form of the given function is

$$F = \bar{x}_1\bar{x}_2 + x_3x_2 + x_1\bar{x}_3x_4$$

The prime implicants of the function are

$\bar{x}_2\bar{x}_3x_4, \bar{x}_1\bar{x}_2, x_3x_2, x_1\bar{x}_3x_4, x_4x_1x_2, \bar{x}_1x_3$

The essential prime implicants are

$$\bar{x}_1\bar{x}_2, x_3x_2, \bar{x}_1\bar{x}_3x_4 \qquad \textbf{Ans.}$$

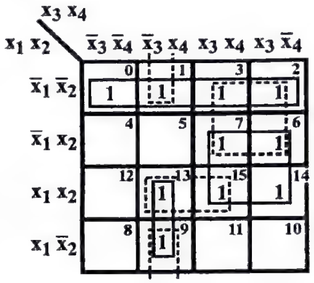


*Fig. 1.50*

***Prob.49. Determine the prime-implicants of the function and minimized function –***

***$F(w, x, y, z) = \Sigma(1, 4, 6, 7, 8, 9, 10, 11, 15)$.***

**(R.G.P.V., Dec. 2010)**

***Or***

***Determine the prime-implicants of the function –***

***$F(w, x, y, z) = \Sigma(1, 4, 6, 7, 8, 9, 10, 11, 15)$***

**(R.G.P.V., June 2017)**

***Sol.*** Given function is

$F(w, x, y, z) = \Sigma(1, 4, 6, 7, 8, 9, 10, 11, 15)$

K-map for given function is shown in fig. 1.51.

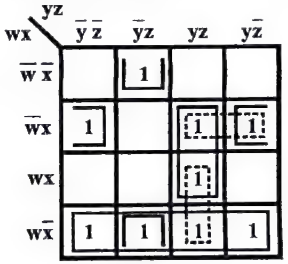


*Fig. 1.51*

From the K-map the prime implicants are given below –

$w\bar{x}, yzx, \bar{y}z\bar{x}, \bar{w}x\bar{z}, y\bar{w}x, wyz$

and minimized function is

$$F = w\bar{x} + yzx + \bar{y}z\bar{x} + \bar{w}x\bar{z} \qquad \textbf{Ans.}$$

***Prob.50. Reduce the expression $F = \Sigma_m(0, 2, 3, 4, 5, 6)$ using K and implement using NAND gates only.*** ***(R.G.P.V., Dec. 2***

***Sol.*** The K-map for the given function is given in fig. 1.52.

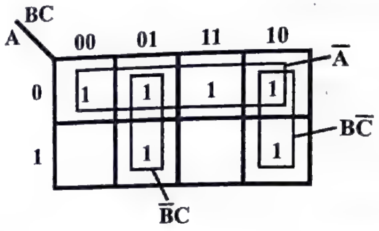


***Fig. 1.52***

The minimized expression is –

$$F = \overline{A} + B\overline{C} + \overline{B}C$$

Implementation of above equation using NAND gate is shown in fig.1

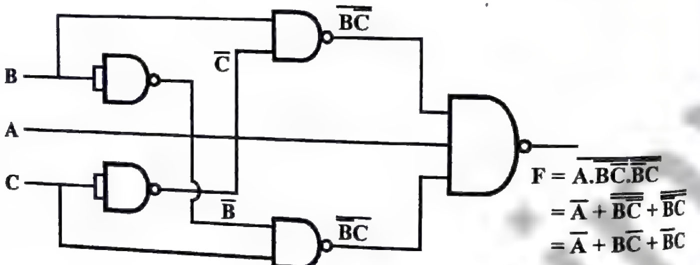

B
A
C
$\overline{C}$
$\overline{B}$
$\overline{B\overline{C}}$
$\overline{\overline{B}C}$
$F = \overline{A.\overline{B\overline{C}}.\overline{\overline{B}C}}$
$= \overline{A} + \overline{\overline{B\overline{C}}} + \overline{\overline{\overline{B}C}}$
$= \overline{A} + B\overline{C} + \overline{B}C$

***Fig. 1.53***

***Prob.51. With the aid of a K-map derive a minimal sum of prod form of –***

***(i) $f(w, x, y, z) = \pi(1, 4, 5, 6, 11, 12, 13, 14, 15)$.***

***Is your answer unique ?***

***(ii) $f(w, x, y, z) = \Sigma(0, 2, 4, 9, 12, 15) + \Sigma\phi(1, 5, 7, 10)$ wher denotes don't care states.***

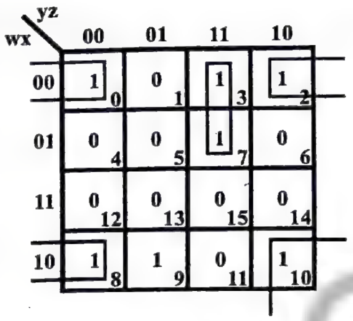
***(R.G.P.V., Dec. 2***

***Sol.*** (i) $f(w, x, y, z) = \pi(1, 4, 5, 6, 11, 12, 13, 14, 15)$

Fig. 1.54 shows the K-map for given function.

From fig. 1.54, the minimal sum of products form is given as follo

$$f = \overline{x}\,\overline{z} + \overline{w}\,yz$$



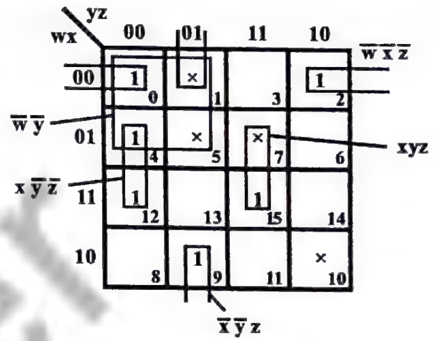

yz
wx
00
01
11
10
$\overline{w}\,\overline{x}\,\overline{z}$
$\overline{w}\,\overline{y}$
xyz
$x\,\overline{y}\,\overline{z}$
$\overline{x}\,\overline{y}\,z$

***Fig. 1.54*** ***Fig. 1.55***

(ii) The K-map for the given expression is shown in fig. 1.55.

$$f = \overline{w}\,\overline{y} + x\overline{y}\,\overline{z} + \overline{x}\,\overline{y}z + xyz + \overline{w}\,\overline{x}\,\overline{z}$$ **Ans.**

***Prob.52. Minimize using Karnaugh map –***

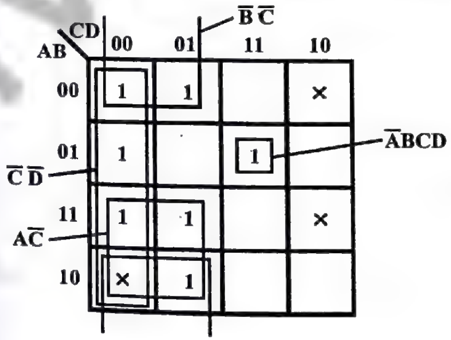
***(i) $\Sigma_m(0, 1, 4, 7, 9, 12, 13) + \Sigma_\phi(2, 8, 14)$***

***(ii) $\Sigma_m(5, 7, 9, 12) + \Sigma_\phi(1, 2, 6, 15)$.***

***(R.G.P.V., June 2012)***

***Sol. (i) $\Sigma_m(0, 1, 4, 7, 9, 12, 13) + \Sigma_\phi(2, 8, 14)$***

The K-map for the above function is shown in fig. 1.56.



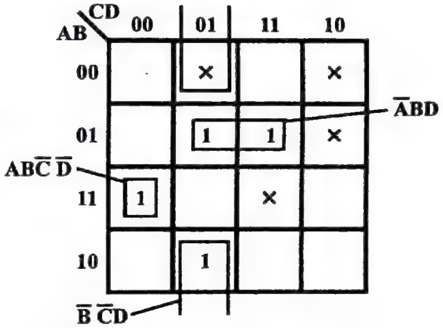


***Fig. 1.56*** ***Fig. 1.57***

The simplified expression is

$$Y = A\overline{C} + \overline{B}\,\overline{C} + \overline{C}\,\overline{D} + \overline{A}BCD$$ **Ans.**

***(ii) $\Sigma_m(5, 7, 9, 12) + \Sigma_\phi(1, 2, 6, 15)$***

The K-map for the above function is shown in fig. 1.57.

The simplified expression is

$$Y = \overline{A}BD + \overline{B}\,\overline{C}D + AB\overline{C}\,\overline{D}$$ **Ans.**

***Prob.53. Plot the following expressions in K-map and then minimize th***

*(i)* $ABCD + A\overline{B}\,\overline{C}\,\overline{D} + A\overline{B}C + AB$

*(ii)* $Y = \Sigma m(7, 9, 10, 11, 12, 13, 14, 15)$.

***(R.G.P.V., Dec. 20***

***Sol.*** (i)

$$Y = ABCD + A\overline{B}\overline{C}\overline{D} + A\overline{B}C + AB$$
$$= ABCD + A\overline{B}\overline{C}\overline{D} + A\overline{B}C(D+\overline{D}) + AB(C+\overline{C})(D+\overline{D})$$
$$= ABCD + A\overline{B}\overline{C}\overline{D} + A\overline{B}CD + A\overline{B}C\overline{D} + (ABC + AB\overline{C})(D+\overline{D})$$
$$= ABCD + A\overline{B}\overline{C}\overline{D} + A\overline{B}CD + A\overline{B}C\overline{D} + ABCD$$
$$+ ABC\overline{D} + AB\overline{C}D + AB\overline{C}\overline{D}$$
$$= ABCD + A\overline{B}\overline{C}\overline{D} + A\overline{B}CD + A\overline{B}C\overline{D} + ABC\overline{D}$$
$$+ AB\overline{C}D + AB\overline{C}\overline{D}$$
$$= m_{15} + m_8 + m_{11} + m_{10} + m_{14} + m_{13} + m_{12}$$
$$\Sigma m(8, 10, 11, 12, 13, 14, 15)$$

K-map for equation (i) is shown in fig. 1.58

Hence, simplified expression is

$$Y = \mathbf{AB + AC + A\overline{D}}$$

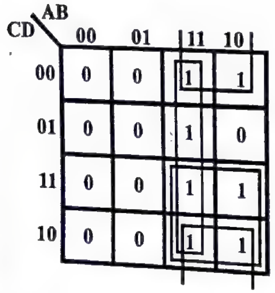


*Fig. 1.58*

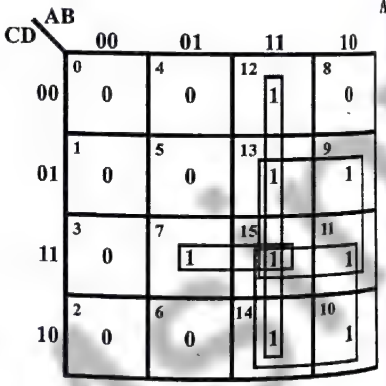


*Fig. 1.59*

(ii) Y = Σm(7, 9, 10, 11, 12, 13, 14, 15)

K-map for equation (ii) is shown in fig. 1.59

Now simplified expression is

$$Y = AB + AD + AC + BCD$$
$$= \mathbf{A(B + D + C) + BCD}$$

***Prob.54. Given the logic equation –***

$$F = \overline{AB}D + AB\overline{C}\,\overline{D} + \overline{A}BD + ABC\overline{D}$$

*(i)* ***Make a truth table***

*(ii)* ***Simplify using K-map.***

***(R.G.P.V., June 20***

***Sol.*** (i) The truth table for the given function is shown in table 1.2

**Table 1.23**

| $A$ | $B$ | $C$ | $D$ | $\overline{A}$ | $\overline{C}$ | $\overline{D}$ | $AB$ | $\overline{AB}$ | $\overline{AB}D$ | $AB\overline{C}\,\overline{D}$ | $\overline{A}BD$ | $ABC\overline{D}$ | $Y$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

(ii) Given function is simplified by using K-map as shown in fig. 1.60.

$$F = \overline{AB}D + AB\overline{C}\,\overline{D} + \overline{A}BD + ABC\overline{D}$$
$$= (\overline{A}+\overline{B}).D + AB\overline{C}\,\overline{D} + \overline{A}BD(C+\overline{C}) + ABC\overline{D}$$
$$= \overline{A}D(B+\overline{B})(C+\overline{C}) + \overline{B}D(A+\overline{A})(C+\overline{C}) + AB\overline{C}\,\overline{D}$$
$$+\overline{A}BCD + \overline{A}B\overline{C}D + ABC\overline{D}$$
$$= (\overline{A}BD + \overline{A}\,\overline{B}D)(C+\overline{C}) + (A\overline{B}D + \overline{A}\,\overline{B}D)(C+\overline{C}) + AB\overline{C}\,\overline{D}$$
$$+ \overline{A}BCD + \overline{A}B\overline{C}D + ABC\overline{D}$$
$$= \overline{A}BCD + \overline{A}B\overline{C}D + \overline{A}\,\overline{B}CD + \overline{A}\,\overline{B}\,\overline{C}D + A\overline{B}CD + A\overline{B}\,\overline{C}D$$
$$+\overline{A}\,\overline{B}CD + \overline{A}\,\overline{B}\,\overline{C}D + AB\overline{CD} + \overline{A}BCD + \overline{A}B\overline{C}D + ABC\overline{D}$$
$$= \overline{A}BCD + \overline{A}B\overline{C}D + \overline{A}\,\overline{B}CD + \overline{A}\,\overline{B}\,\overline{C}D + A\overline{B}CD + A\overline{B}\,\overline{C}D$$
$$+AB\overline{C}\,\overline{D} + ABC\overline{D}$$
$$= \Sigma m = (1, 3, 5, 7, 9, 11, 12, 14)$$

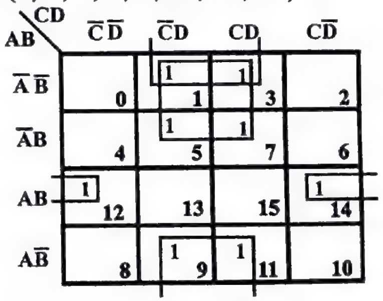


*Fig. 1.60*

The simplified expression is

$$F = \mathbf{\overline{A}D + \overline{B}D + AB\overline{D}}$$ **Ans.**

**Prob.53. Plot the following expressions in K-map and then minimize th**

*(i)* $ABCD + A\overline{B}\,\overline{C}\,\overline{D} + A\overline{B}C + AB$

*(ii)* $Y = \Sigma m(7, 9, 10, 11, 12, 13, 14, 15)$.

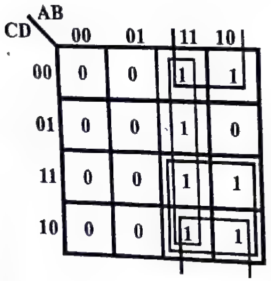
**(R.G.P.V., Dec. 20**

***Sol.*** (i)
$$\begin{aligned} Y &= ABCD + A\overline{B}\,\overline{C}\,\overline{D} + A\overline{B}C + AB \\ &= ABCD + A\overline{B}\,\overline{C}\,\overline{D} + A\overline{B}C(D+\overline{D}) + AB(C+\overline{C})(D+\overline{D}) \\ &= ABCD + A\overline{B}\,\overline{C}\,\overline{D} + A\overline{B}CD + A\overline{B}C\overline{D} + (ABC + AB\overline{C})(D+ \\ &= ABCD + A\overline{B}\,\overline{C}\,\overline{D} + A\overline{B}CD + A\overline{B}C\overline{D} + ABCD \\ &\qquad + ABC\overline{D} + AB\overline{C}D + AB\overline{C}\,\overline{D} \\ &= ABCD + A\overline{B}\,\overline{C}\,\overline{D} + A\overline{B}CD + A\overline{B}C\overline{D} + ABC\overline{D} \\ &\qquad + AB\overline{C}D + AB\overline{C}\,\overline{D} \\ &= m_{15} + m_8 + m_{11} + m_{10} + m_{14} + m_{13} + m_{12} \\ &\qquad \Sigma m(8, 10, 11, 12, 13, 14, 15) \end{aligned}$$

K-map for equation (i) is shown in fig. 1.58

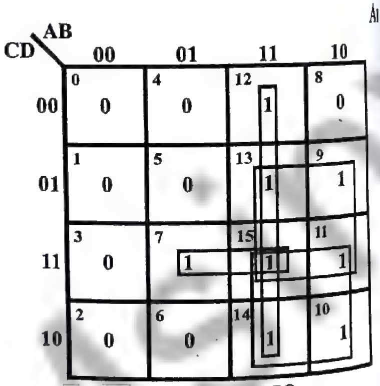
Hence, simplified expression is

$$Y = \mathbf{AB + AC + A\overline{D}}$$



| CD \ AB | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 1 | 1 |
| 01 | 0 | 0 | 1 | 0 |
| 11 | 0 | 0 | 1 | 1 |
| 10 | 0 | 0 | 1 | 1 |

*Fig. 1.58*



| CD \ AB | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 1 | 0 |
| 01 | 0 | 0 | 1 | 1 |
| 11 | 0 | 1 | 1 | 1 |
| 10 | 0 | 0 | 1 | 1 |

*Fig. 1.59*

(ii) $Y = \Sigma m(7, 9, 10, 11, 12, 13, 14, 15)$

K-map for equation (ii) is shown in fig. 1.59

Now simplified expression is

$$\begin{aligned} Y &= AB + AD + AC + BCD \\ &= \mathbf{A(B + D + C) + BCD} \end{aligned}$$

**Prob.54. Given the logic equation –**

$$F = \overline{AB}D + AB\overline{C}\,\overline{D} + \overline{A}BD + ABC\overline{D}$$

*(i)* ***Make a truth table***

*(ii)* ***Simplify using K-map.***

**(R.G.P.V., June 20**

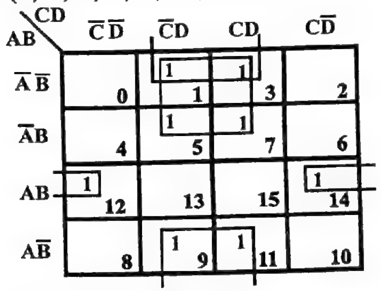
***Sol.*** (i) The truth table for the given function is shown in table 1.23

**Table 1.23**

| $A$ | $B$ | $C$ | $D$ | $\overline{A}$ | $\overline{C}$ | $\overline{D}$ | $AB$ | $\overline{AB}$ | $\overline{AB}D$ | $AB\overline{C}\,\overline{D}$ | $\overline{A}BD$ | $ABC\overline{D}$ | $Y$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

(ii) Given function is simplified by using K-map as shown in fig. 1.60.

$$\begin{aligned} F &= \overline{AB}D + AB\overline{C}\,\overline{D} + \overline{A}BD + ABC\overline{D} \\ &= (\overline{A} + \overline{B}).D + AB\overline{C}\,\overline{D} + \overline{A}BD(C+\overline{C}) + ABC\overline{D} \\ &= \overline{A}D(B+\overline{B})(C+\overline{C}) + \overline{B}D(A+\overline{A})(C+\overline{C}) + AB\overline{C}\,\overline{D} \\ &\qquad + \overline{A}BCD + \overline{A}B\overline{C}D + ABC\overline{D} \\ &= (\overline{A}BD + \overline{A}\,\overline{B}D)(C+\overline{C}) + (A\overline{B}D + \overline{A}\,\overline{B}D)(C+\overline{C}) + AB\overline{C}\,\overline{D} \\ &\qquad + \overline{A}BCD + \overline{A}B\overline{C}D + ABC\overline{D} \\ &= \overline{A}BCD + \overline{A}B\overline{C}D + \overline{A}\,\overline{B}CD + \overline{A}\,\overline{B}\,\overline{C}D + A\overline{B}CD + A\overline{B}\,\overline{C}D \\ &\qquad + \overline{A}\,\overline{B}CD + \overline{A}\,\overline{B}\,\overline{C}D + AB\overline{CD} + \overline{A}BCD + \overline{A}B\overline{C}D + ABC\overline{D} \\ &= \overline{A}BCD + \overline{A}B\overline{C}D + \overline{A}\,\overline{B}CD + \overline{A}\,\overline{B}\,\overline{C}D + A\overline{B}CD + A\overline{B}\,\overline{C}D \\ &\qquad + AB\overline{C}\,\overline{D} + ABC\overline{D} \\ &= \Sigma m = (1, 3, 5, 7, 9, 11, 12, 14) \end{aligned}$$



| AB \ CD | $\overline{C}\,\overline{D}$ | $\overline{C}D$ | $CD$ | $C\overline{D}$ |
|---|---|---|---|---|
| $\overline{A}\,\overline{B}$ |  | 1 | 1 |  |
| $\overline{A}B$ |  | 1 | 1 |  |
| $AB$ | 1 |  |  | 1 |
| $A\overline{B}$ |  | 1 | 1 |  |

*Fig. 1.60*

The simplified expression is

$$F = \mathbf{\overline{A}D + \overline{B}D + AB\overline{D}}$$ **Ans.**

***Prob.55. Realise the following function as –***
***(i) Multilevel NAND-NAND gate network and***
***(ii) Multilevel NOR-NOR network.***

$$F = A\overline{B}C + B(C+\overline{D}) + \overline{AD}$$

*Sol.* The given function can be implemented as a four level AND-OR gate network as shown in fig. 1.61.

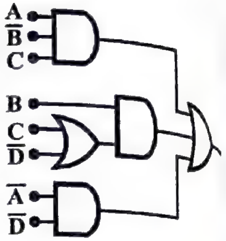


*Fig. 1.61*

***(i) Multilevel NAND-NAND Implementation –***

(a) Each OR gate is replaced by OR gate with bubbles at its inpu

(b) Each AND gate is replaced by a NAND gate followed by inverter.

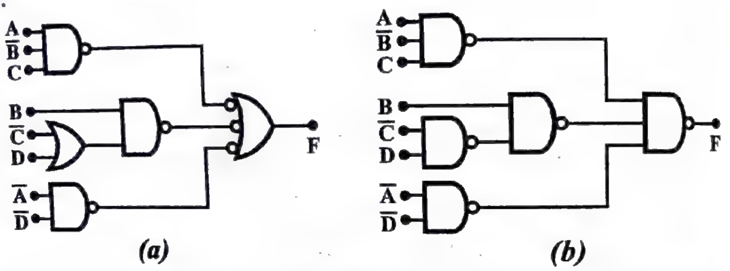


*Fig. 1.62*

The corresponding input variables to bubbled OR gate is complement so that the logically equivalent of the function is maintained as shown in fig 1.62 (a). Now the logic circuit shown in fig. 1.62 (a) can be modified shown in fig. 1.62 (b).

***(ii) Multilevel NOR-NOR Implementation –***

(a) Each AND gate is replaced by AND gate with bubbles at its input

(b) Each OR gate is replaced by a NOR gate followed by an invert

The corresponding input variables to bubbled AND gate is complement so that the logically equivalent of the function is maintained as shown in fig 1.63 (a). Now the logic circuit shown in fig. 1.63 (a) can be modified shown in fig. 1.63 (b).

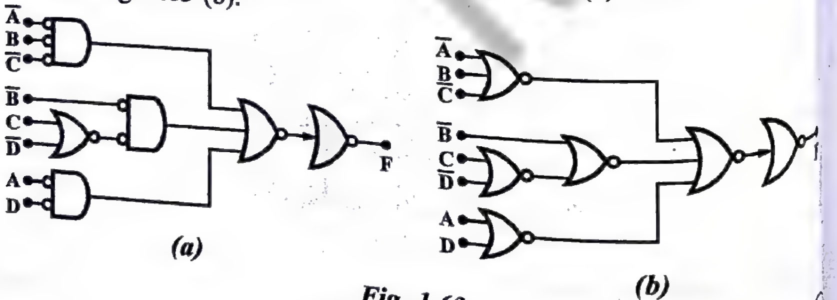


*Fig. 1.63*

# UNIT 2

## COMBINATIONAL LOGIC – HALF ADDER, HALF SUBTRACTOR, FULL ADDER, FULL SUBTRACTOR

***Q.1. What is meant by a combinational logic circuit ?***

*Ans.* In combinational logic circuits, the outputs at any instant of time depend upon the inputs present at that instant of time. A combinational circuit consists of logic gates, they performs a specific information processing operation fully specified logically by a set of Boolean function. Fig. 2.1 shows a block diagram of such combinational logic system, with n input binary variables come from an external source, the m output variables go to an external destination.

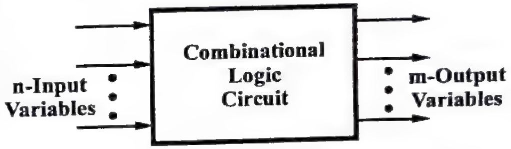


*Fig. 2.1 Block Diagram of a Combinational Logic Circuit*

For an input variable, there are $2^n$ possible combinations of binary inputs. For each possible input combination, there is one and only one possible output combination.

A combinational circuit can be described by m Boolean functions, one for each output variable. Each output function is expressed in terms of the n input variables.

***Q.2. Explain the design procedure of combinational circuits.***

*Ans.* The design of combinational circuits starts from the specification of the problem and culminates in a logic circuit diagram or a set of Boolean functions from which the logic diagram can be obtained. The procedure involves the following steps –

(i) From the specifications of the circuit, determine the required number of inputs and outputs and assign a symbol to each.

(ii) Derive the truth table that defines the required relationship between inputs and outputs.

(iii) Obtain the simplified Boolean functions for each output a function of the inp ıt variables.

(iv) Draw the logic diagram and verify the correctness of the desig

A truth table for a combinational circuit consists of input columns and outp columns. The input columns are obtained from the $2^n$ binary numbers for th input variables. The binary values for the outputs are determined from the stat specifications. The output functions specified in the truth table give the exa definition of the combinational circuit. It is important that the verbal specificatio be interpreted correctly in the truth table. Word specifications are often incomple and any wrong interpretation may result in an incorrect truth table.

***Q.3. Explain the working of half adder.*** ***(R.G.P.V., Dec. 201***

***Or***

***Draw the circuit diagram for half adder.*** ***(R.G.P.V., June 201***

***Or***

***Write short note on half adder.*** ***(R.G.P.V., Dec. 201***

***Or***

***Design a half adder.*** ***(R.G.P.V., Dec. 201***

***Ans.*** A half-adder is a combinational circuit, which performs the arithmet addition of two binary digits, giving a sum bit and a carry bit. The truth table o an half-adder is given in table 2.1. The output sum (S) bit is the EX-OR operatio of two inputs A and B. Therefore,

$$S = \overline{A}B + A\overline{B} = A \oplus B$$

**Table 2.1 Truth Table of a Half-adder**

| *Inputs* | | *Outputs* | |
|---|---|---|---|
| *Augend (A)* | *Addend (B)* | *Sum (S)* | *Carry (C)* |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

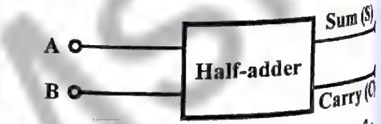


***Fig. 2.2 Block Diagram of a Half-adder***

The output carry (C) bit is the AND operation of two inputs A and B. Thu

$$C = A.B$$

The block diagram of an half-adder is shown in fig. 2.2.

The half-adder circuit can be implemented by using EX-OR and AND gates as shown in fig. 2.3.

The half-adder realization using NAND gates is shown in fig. 2.4.

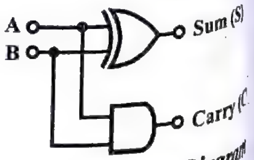


***Fig. 2.3 Logic Diagram of a Half-adder***

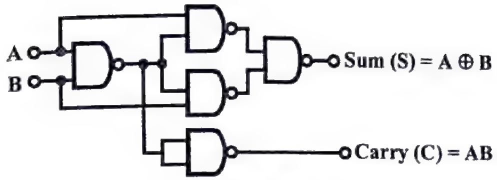


***Fig. 2.4 Realization of a Half-adder Using NAND Gates***

***Q.4. Explain half subtractor circuit.*** ***(R.G.P.V., Dec. 2013)***

***Ans.*** A half-subtractor is an arithmetic circuit which is used to perform subtraction of two bits. It has two inputs minuend (A), subtrahend (B) and two outputs difference bit (D) and borrow out bit ($B_{out}$). The block diagram of half-subtractor is given in fig. 2.5 and its truth table is shown in table 2.2.

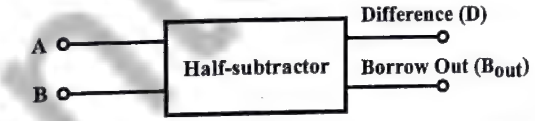


***Fig. 2.5 Block Diagram of a Half-subtractor***

**Table 2.2 Truth Table of a Half-subtractor**

| *Inputs* | | *Outputs* | |
|---|---|---|---|
| *Minuend (A)* | *Subtrahend (B)* | *Difference (D)* | *Borrow Out ($B_{out}$)* |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |

A half-subtractor is used to subtract the least significant bit (LSB) of the subtrahend from the least significant bit (LSB) of the minuend when one binary number is subtracted from the other.

From the truth table, it is clear that the difference output is 0 (or LOW) when both the inputs are same, i.e., (A = B) and difference output is 1 when both the inputs are in complemented form, i.e., (A ≠ B). If the input A is less than input B i.e., (A < B), then subtraction is done by borrowing 1 from the next higher order bit. The logic diagram of half-subtractor is shown in fig. 2.6.

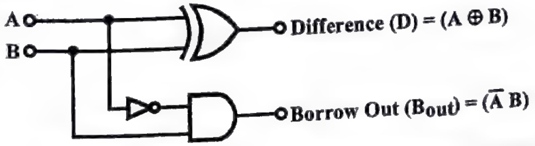


***Fig. 2.6 Logic Diagram of a Half-subtractor***

A circuit that produces the correct difference and borrow bits in re to every possible combination of the two 1-bit number is described as

$$D = \overline{A}B + A\overline{B} = A \oplus B$$

$$B_{out} = \overline{A}B$$

*Q.5. Design a half subtractor using NAND gates.*

*Ans.* The NAND logic implementation of half-subtractor is given

$$D = A \oplus B$$

$$= \overline{\overline{A.\overline{AB}}.\overline{B.\overline{AB}}}$$

$$B_{out} = \overline{A}B = B(\overline{A} + \overline{B})$$

$$= B(\overline{AB}) = \overline{\overline{B.\overline{AB}}}$$

The logic diagram of half-subtractor using NAND gates is depicted in fig. 2.7.

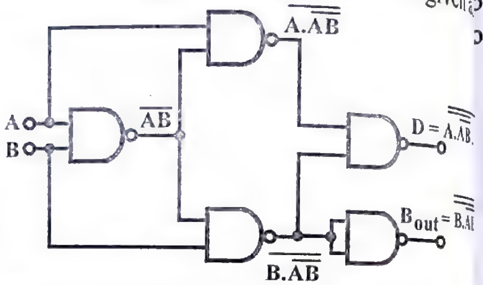

$\overline{A.\overline{AB}}$
A
B
$\overline{AB}$
$D = \overline{\overline{A.\overline{AB}}...}$
$B_{out} = \overline{\overline{B.A...}}$
$\overline{B.\overline{AB}}$

*Fig. 2.7 Realization of Half-subtractor U NAND Gates*

***Q.6. Design a half subtractor using NOR gates.***

***Ans.*** The NOR logic implementation of half-subtractor is given as-

$$D = A \oplus B = \overline{A}B + A\overline{B} = \overline{A}B + A\overline{A} + A\overline{B} + B\overline{B}$$

$$= \overline{A}(A+B) + \overline{B}(A+B) = \overline{A + \overline{A+B}} + \overline{B + \overline{A+B}}$$

$$B_{out} = \overline{A}B = \overline{A}(A+B) = \overline{\overline{\overline{A}(A+B)}} = \overline{A + \overline{(A+B)}}$$

The logic diagram of half-subtractor using NOR gates is depicte fig. 2.8.

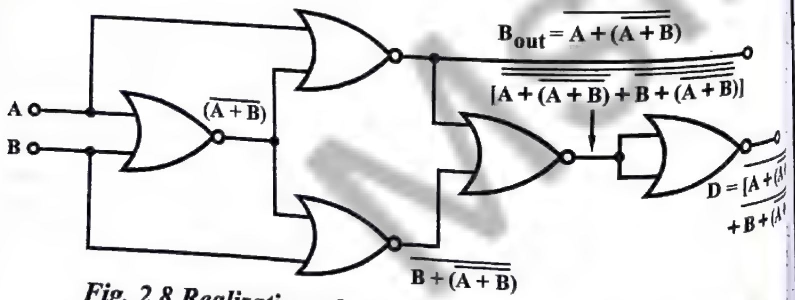

$B_{out} = \overline{A + \overline{(A+B)}}$
$[\overline{\overline{A + \overline{(A+B)}} + \overline{B + \overline{(A+B)}}}]$
A
B
$(\overline{A+B})$
$D = [\overline{A + (A}$
$\overline{+B+(A}$
$\overline{B + \overline{(A+B)}}$

***Fig. 2.8 Realization of a Half-subtractor using NOR Gates***

***Q.7. Design a full-adder with the help of truth table. Explain the wor of full-adder by giving expressions for sum and carry in full-adder.***

***(R.G.P.V., June 2***

***Or***

***Explain full adder.*** ***(R.G.P.V., Dec. 2***

***Or***

***Draw the truth table and logic diagram of full adder. (R.G.P.V., Dec. 2015)***

***Or***

***Design and draw a full adder circuits.*** ***(R.G.P.V., Dec. 2017)***

***Ans.*** A full-adder is a device capable of performing binary addition of three binary digits, such as A, B and carry-in ($C_{in}$) and generates a sum output and an output carry ($C_{out}$). The block diagram of full-adder is shown in fig. 2.9.

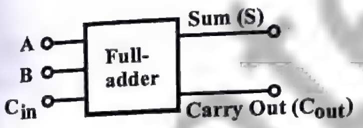


***Fig. 2.9 Block Diagram of a Full-adder***

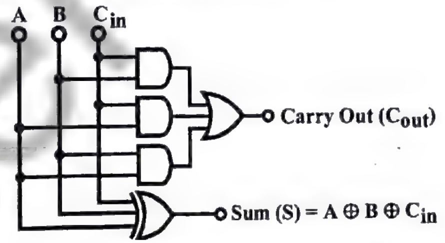


***Fig. 2.10 Realization of a Full-adder Using Various Logic Gates***

The realization of full-adder using three input EX-OR gate, three two input AND gates and one three input OR gate is shown in fig. 2.10.

The truth table for the full-adder circuit is shown in table 2.3. The binary variable S gives the value of the least significant bit of the sum and the binary variable $C_{out}$ gives the output carry.

**Table 2.3 Truth Table of Full-adder**

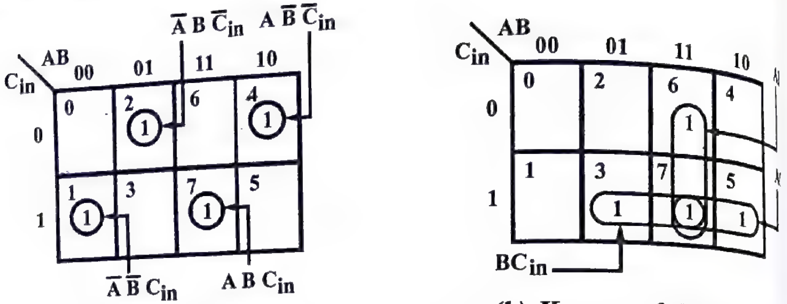
| *Inputs* | | *Outputs* | | |
|---|---|---|---|---|
| ***Augend Bit (A)*** | ***Addend Bit (B)*** | ***Carry-in ($C_{in}$)*** | ***Sum (S)*** | ***Carry out ($C_{out}$)*** |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

The K-map for the output S and $C_{out}$ are given in figs. 2.11 (a) and (b), respectively.

The simplified logic expression for output S and $C_{out}$ can be written as

$$S = \overline{A}B\overline{C}_{in} + \overline{A}\,\overline{B}C_{in} + A\overline{B}\,\overline{C}_{in} + ABC_{in} \quad ...(i)$$

$$C_{out} = AB + BC_{in} + AC_{in} \quad ...(ii)$$


$\overline{A}B\overline{C}_{in}$
$A\overline{B}\,\overline{C}_{in}$
$C_{in}$
AB
00
01
11
10
0
1
$\overline{A}\,\overline{B}C_{in}$
$ABC_{in}$
$BC_{in}$

*(a) K-map of Output S* *(b) K-map of Output $C_{out}$*

*Fig. 2.11*

***Q.8. How to design a combination circuit ? Design a full adder ch***

**(R.G.P.V., May**

***Ans.*** Refer the ans. of Q.2 and Q.7.

***Q.9. Design a full adder using NAND gates.***

***Ans.*** The NAND logic implementation of full adder is given as

$$S = \overline{A}B\overline{C}_{in} + \overline{A}\,\overline{B}C_{in} + A\overline{B}\,\overline{C}_{in} + ABC_{in}$$

$$C_{out} = AB + BC_{in} + AC_{in}$$

The NAND gates realization of the expression of S and $C_{out}$ in equ (i) and (ii) is shown in figs. 2.12 and 2.13, respectively.

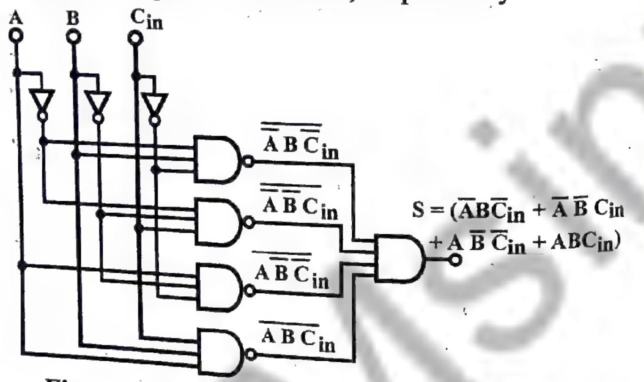

A
B
$C_{in}$
$\overline{\overline{A}B\overline{C}_{in}}$
$\overline{\overline{A}\,\overline{B}C_{in}}$
$\overline{A\overline{B}\,\overline{C}_{in}}$
$\overline{ABC_{in}}$
$S = (\overline{A}B\overline{C}_{in} + \overline{A}\,\overline{B}\,C_{in} + A\,\overline{B}\,\overline{C}_{in} + ABC_{in})$

*Fig. 2.12 NAND Gates Realization of Output S*

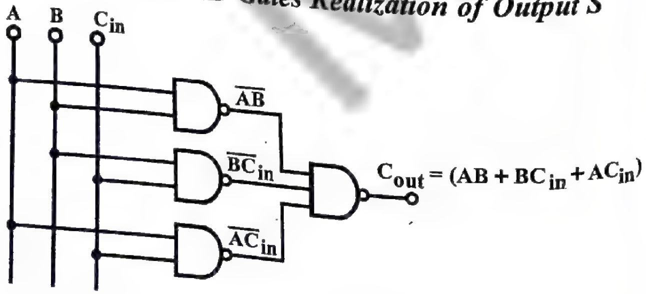


*Fig. 2.13 NAND Gates Realization of Output $C_{out}$*

***Q.10. Implement a full-adder circuit using two half-adders and an OR gate.*** ***(R.G.P.V., Feb. 2010)***

***Or***

***Design a full-adder with two half-adders and an OR gate.*** ***(R.G.P.V., Dec. 2010)***

***Ans.*** The full-adder must add the two input bits and the input carry. For the input carry ($C_{in}$) to be added to the input bits, it must be Ex-ORed with A ⊕ B, giving the equation for the sum output of the full-adder as –

$$S = (A \oplus B) \oplus C_{in}$$

This means that to implement the full-adder sum function, two EX-OR gates of two inputs can be used. The first must generate the term A ⊕ B and the second has its inputs the output of the first EX-OR gate and the input carry ($C_{in}$), as shown in fig. 2.14.

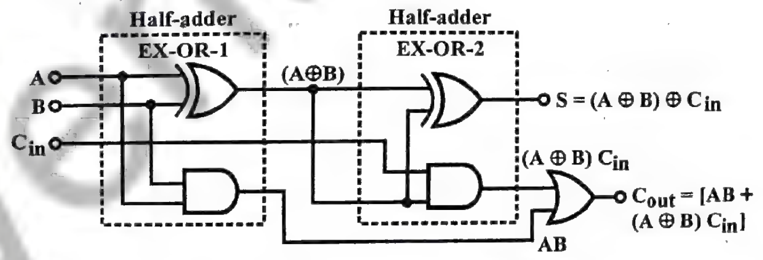


*Fig. 2.14 Logic Diagram of a Full-adder Using Two Half-adder and OR Gate*

The output carry ($C_{out}$) is a 1 when both inputs to the first EX-OR gate are 1's or when both inputs to the second EX-OR gate are 1's. The output carry ($C_{out}$) of the full-adder is therefore produced by the inputs A ANDed with B and A ⊕ B ANDed with $C_{in}$. These two terms are ORed as given below –

$$C_{out} = AB + (A \oplus B)\ C_{in}$$

Even though a full-adder can be constructed using two half-adders and one OR gates, the disadvantage is that the bits must propagate through various gates in succession, which makes the total propagation delay greater than that of the full-adder circuit using AND-OR-Invert logic.

***Q.11. Implement a full adder circuit with a decoder and two OR gates.*** ***(R.G.P.V., June 2014)***

***Or***

***Implement a full adder circuit with a (3 to 8 lines) decoder and two OR gates.*** ***(R.G.P.V., Dec. 2017)***

***Ans.*** The procedure for realizing a combinational circuit by means of a decoder and OR gates needs that the Boolean function for the circuit is expressed in sum of minterms. A decoder is then selected to generates all the minterms of the input variables. The input to each OR gate are selected from the decod outputs according to the list of minterm of each function. This procedure w be shown by an example that implements a full-adder circuit. For the tru table of full-adder circuit, refer table 2.3. From this table, we obtain the functio for the combinational circuit in sum of minterms as –

$$S\,(A, B, C_{in}) = \Sigma\, m\,(1, 2, 4, 7)$$

$$C_{out}\,(A, B, C_{in}) = \Sigma\, m\,(3, 5, 6, 7)$$

Since there are three inputs and a total of eight minterms, so we requir 3 to 8 decoder. The implementation of a full-adder with 3 to 8 decoder and O gates is shown in fig. 2.15. The 3 to 8 decoder generates the eight minter for three inputs A, B, $C_{in}$. The OR gate for output S forms the logical sum minterms 1, 2, 4 and 7. The OR gate for output C forms the logical sum minterms 3, 5, 6 and 7.

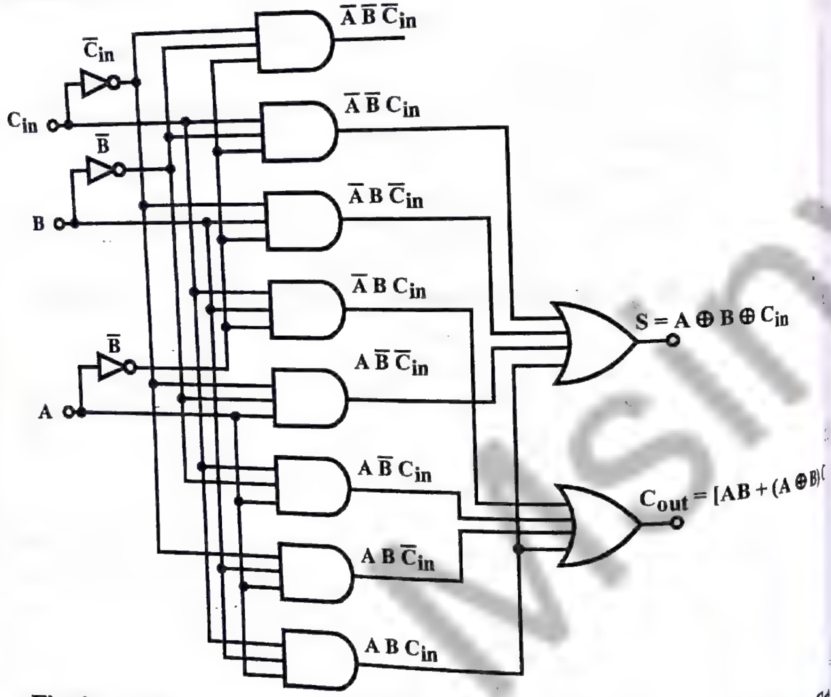

$\bar{C}_{in}$
$C_{in}$
$\bar{B}$
B
$\bar{B}$
A
$\bar{A}\,\bar{B}\,\bar{C}_{in}$
$\bar{A}\,\bar{B}\,C_{in}$
$\bar{A}\,B\,\bar{C}_{in}$
$\bar{A}\,B\,C_{in}$
$A\,\bar{B}\,\bar{C}_{in}$
$A\,\bar{B}\,C_{in}$
$A\,B\,\bar{C}_{in}$
$A\,B\,C_{in}$
$S = A \oplus B \oplus C_{in}$
$C_{out} = [AB + (A \oplus B)C$

*Fig. 2.15 Implementation of a Full-adder with 3 to 8 Decoder and OR Ga*

A function with a long list of minterms needs an OR gate with a la number of inputs. A function having a list of p minterms can be expressed its complement form $\overline{F}$ with $2^n - p$ minterms. When the number of minter in a function is greater than $2^n/2$, then $\overline{F}$ can be expressed some minterms such a case, it is advantageous to use a NOR gate to sum the minterms of The output of the NOR gate inverts this sum and generates the output F.

***Q.12. Explain full adder and design a full adder circuit using 3 to 8 decoder and two OR gates.*** ***(R.G.P.V., Dec. 2016)***

***Ans.*** Refer the ans. of Q.7 and Q.11.

***Q.13. Implement/Design a full-subtractor using two half-subtractor and an OR gate.*** ***(R.G.P.V., June 2005, 2010, 2015)***

***Or***

***Design a full subtractor circuit.*** ***(R.G.P.V., June 2012)***

***Ans.*** A full-subtractor is a combinational circuit which performs subtraction involving binary bits, such as minuend bit, subtrahend bit and the borrow from the previous stage. The block diagram of full-subtractor is given in fig. 2.16 and its truth table is shown in table 2.4.

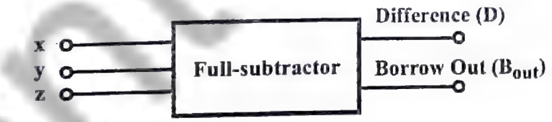


***Fig. 2.16 Block Diagram of a Full-subtractor***

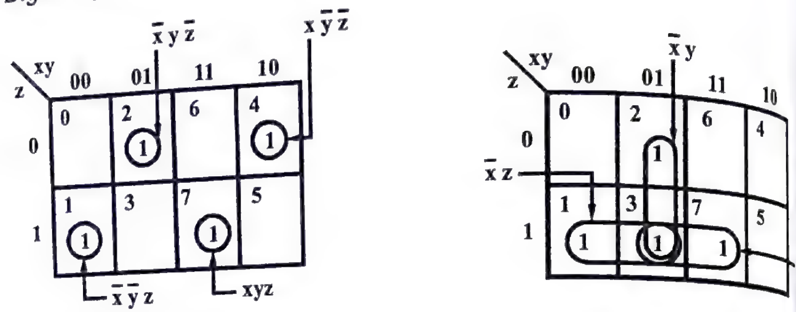
**Table 2.4 Truth Table of a Full-subtractor**

| *Inputs* | | | *Outputs* | |
|---|---|---|---|---|
| *Minuend (x)* | *Subtrahend (y)* | *Previous Stage Borrow (z)* | *Difference (D)* | *Borrow out ($B_{out}$)* |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

The K-maps for difference (D) and borrow (B) are shown in figs. 2.17 (a) and (b), respectively.

The simplified expressions for both the K-maps can be written as –

$$\begin{aligned} D &= \bar{x}y\bar{z} + x\bar{y}\,\bar{z} + xyz + \bar{x}\,\bar{y}z \\ &= \bar{z}\,(\bar{x}y + x\bar{y}) + z(xy + \bar{x}\,\bar{y}) \\ &= \bar{z}(x \oplus y) + z\,(\overline{x \oplus y}) = x \oplus y \oplus z \\ B_{out} &= \bar{x}y + \bar{x}z + yz = \bar{x}y + z(\overline{x \oplus y}) \end{aligned}$$



*(a) K-map for Output D*

*(b) K-map for Output B*

*Fig. 2.17*

The implementation of full-subtractor using two half-subtractor and gate is shown in fig. 2.18.

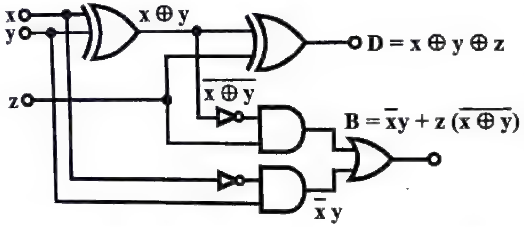


*Fig. 2.18 Implementation of a Full-subtractor using Two Half-subtractor and OR Gate*

*Q.14. Discuss full-subtractor and full-adder.* *(R.G.P.V., June 20*

*Ans.* **Full Subtractor** – Refer the ans. of Q.13.

**Full-adder** – Refer the ans. of Q.7.

*Q.15. Show how a full adder can be converted to a full subtractor w the addition of one inverter circuit.* *(R.G.P.V., June 20*

*Ans.* Refer the ans. of Q.13.

It is clear that D is the same as the sum output for a full adder, and the borrow output $B_{out}$ resembles the carry output for full adder except that one of the inputs is complemented. From these similarities, it is possible to convert a full adder into a full subtractor by merely complementing that input prior to its application to the input of gates which form the borrow output.

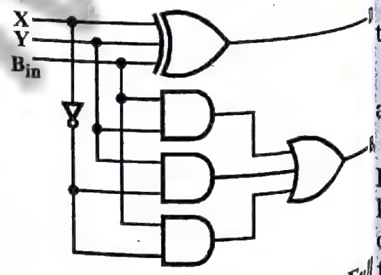


*Fig. 2.19 Realization of Full Subtractor*

*Q.16. Implement full-subtractor using any logic gate.* *(R.G.P.V., June 2009)*

*Or*

*Design a full subtractor using logic gates.* *(R.G.P.V., Dec. 2014)*

*Ans.* The full-subtractor produces the difference (D) and borrow (B) bits in response to every possible combination of x, y and z, which is given by

$$D = x \oplus y \oplus z = \overline{\overline{(x \oplus y)\,\overline{(x \oplus y)\,z}}.\overline{z\overline{(x \oplus y)\,z}}}$$

and $$B_{out} = \bar{x}y + z(\overline{x \oplus y}) = \overline{\overline{\bar{x}y + z(\overline{x \oplus y})}}$$

$$= \overline{\overline{\bar{x}y}.\overline{z(\overline{x \oplus y})}} = \overline{\overline{y(\bar{x}+\bar{y})}.\overline{z[\bar{z}+\overline{(x \oplus y)}]}} = \overline{\overline{y.\overline{xy}}.\overline{z[\overline{z.(x \oplus y)}]}}$$

By using Boolean expressions of difference D and borrow B, we can easily implement the logic circuit of full-subtractor having only NAND gates as shown in fig. 2.20.

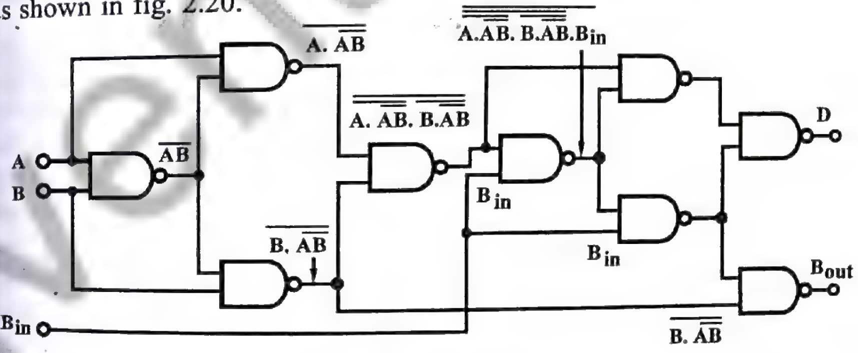

A. $\overline{AB}$
$\overline{\overline{A.\overline{AB}}.\overline{B.\overline{AB}}.B_{in}}$
$\overline{\overline{A.\overline{AB}}.\overline{B.\overline{AB}}}$
$\overline{AB}$
A
B
$B_{in}$
B. $\overline{AB}$
$B_{in}$
D
$B_{out}$
$B_{in}$
B. $\overline{AB}$

*Fig. 2.20 Implementation of Full-subtractor Using NAND Logic Gates*

*Q.17. Design a full subtractor using minimum logic gates. Also design the circuit using all NAND gates.* *(R.G.P.V., Dec. 2012)*

*Sol.* **Full Subtractor using Minimum Logic Gates** – Refer the ans. of Q.13.

**Full Subtractor Circuit using all NAND Gates** – Refer the ans. of Q.16.

*Q.18. Design a full subtractor circuit using decoder and OR gates.* *(R.G.P.V., Dec. 2015)*

*Ans.* From the truth table of a full subtractor (see table 2.4), we obtain the function for this combinational circuit in sum of minterms.

$$D(x, y, z) = \Sigma(1, 2, 4, 7)$$

and $$B_{out}(x, y, z) = \Sigma(1, 2, 3, 7)$$

A full subtractor circuit using decoder and OR gates is shown in fig. 2.21. Because there are three inputs and total of eight minterms, we require a 3 to 8 line decoder. Decoder generates the 8 minterms for x, y, z. The OR gate for output D forms the sum of minterms 1, 2, 4, 7 and the OR gate for output $B_{out}$ froms the sum of minterms 1, 2, 3, 7.

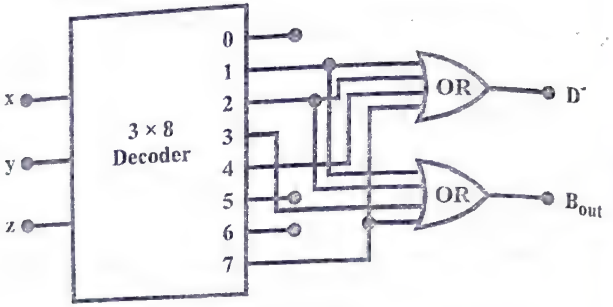


Fig. 2.21

## SERIES AND PARALLEL ADDITION, LOOK-AHEAD CAR GENERATOR, BCD ADDER

***Q.19. Describe serial addition technique.***

***Ans.*** In serial adder, the addition operation is carried out bit by bit the serial adder needs simple circuitry compared to parallel adder. Initial addend and augend each are loaded into an n-bit shift register. To recei sum an (n + 1) bit shift register is available. The register shift to right a register loading is such that the least significant bits (LSBs) $A_0$ and $B_0$ the right most register positions. Initially, we also assume that the D flip-flop is in the reset state, so that $Q_0 = C_k = 0$. The sum bits will be s into the register from the left, again 1-bit at a time. There is no need to the sum register at the outset of the addition sequence.

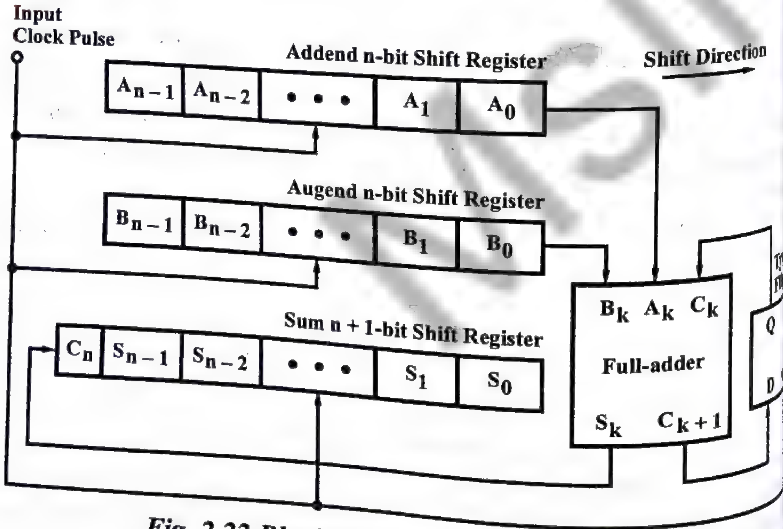


*Fig. 2.22 Block Diagram of a 4-bit Serial Adder*

Initially, before the first triggering edge of the clock waveform, the bits $A_0$ nd $B_0$ will be present at the full-adder inputs $A_k$ and $B_k$ while $C_k$ input will be zero. At that time, the sum $S_0$ and $C_0$ generated by these bits will be present at he full-adder outputs $S_k$ and $C_{k+1}$. A 4-bit serial adder is shown in fig. 2.22.

At the first triggering clock edge, a number of events will occur. The sum bit $S_0$ will register in the leftmost flip-flop of the sum register. The addend and augend registers will shift one position so that the bits of next higher order $A_1$ and $B_1$ will now be input to the full-adder. The $C_1$ input to the D-type flip-flop will be transferred to its output so that $C_1$ will be present at the full-adder input $C_k$. After this first clock edge, the first sum bit has been registered, the input to the full-adder are $A_1$, $B_1$ and $C_1$, the outputs of full-adder are $S_1$ and $C_2$. The second clock edge moves $S_0$ one place to the right in the sum register and registers the newly calculated sum bit $S_1$ in the leftmost flip-flop of the sum register.

The second clock edge advances $A_2$ and $B_2$ to the full-adder and also transfer $C_2$ to the input $C_k$. In this manner, the sum bits are calculated, entered into the sum register from the left and shifted down the register to make room for successively calculated bits. After n clock edges the registers A and B will be cleared, we shall take $A_k = B_k = 0$ and $C_k$ will be the carry $C_n$ generated by the addition $A_{k-1} + B_{k-1} + C_{k-1}$. The (n + 1)th clock edge will transfer last bit into the sum register and the operation will be complete.

***Q.20. Explain the working of a 4-bit parallel binary adder.***
***(R.G.P.V., June 2008, 2009)***

***Or***

***Describe the design of parallel adder. (R.G.P.V., Dec. 2008)***

***Or***

***Design a 4-bit adder with carry look ahead. (R.G.P.V., Feb. 2010)***

***Ans.*** A basic 4-bit parallel adder is implemented with four full-adder stages as given in fig. 2.23. The least significant bits ($A_0$ and $B_0$) in each number being added to go into the right most full-adder, the higher order bits are applied as given to the successively higher order adders. With the most significant bits ($A_3$ and $B_3$) in each number being applied to the left most full-adder. The carry output of each adder is connected to the carry input of the next higher order adder as shown in fig. 2.23.

There are two types of parallel adder in terms of the method used to handle carries, namely ripple carry adder and the look-ahead carry adder. The carry output of each full-adder is connected to the carry input of the next higher order stage, then the adder is known as ripple carry adder. The method of speeding up the addition process by eliminating this ripple carry delay is known as look-ahead carry addition. The look-ahead carry adder anticipates the output carry of each stage and based on the input bits of each stage, produces the output carry by either carry generation or carry propagation.

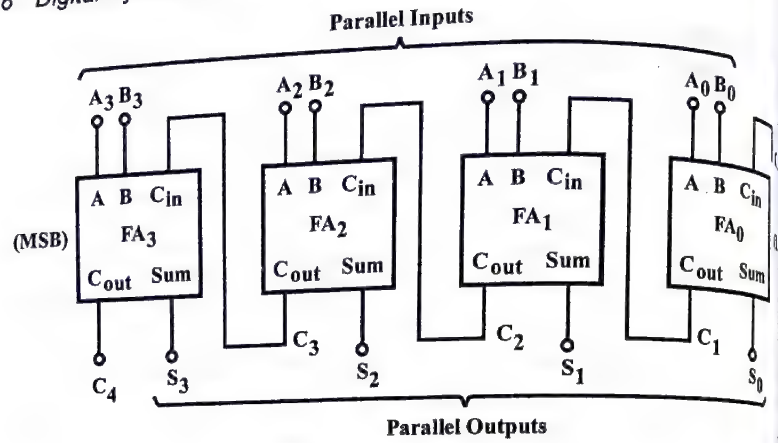


*Fig. 2.23 Block Diagram of a 4-bit Binary Parallel Adder*

The carry generation occurs when an output carry is generated inte by the full-adder. A carry is generated only when both input bits are 1's carry generated $C_g$ is expressed as the AND function of the two input and B as –

$$C_g = AB$$

The carry propagation ($C_p$) occurs when the input carry is ripp become the output carry. An input carry may be propagated by the full when either or both of the input bits are 1's. The propagated carry ( expressed as the OR function of the input bits.

$$C_p = A \oplus B$$

The truth table of a 4-bit binary parallel adder is given in table 2.5 subscript n represents the adder bits and can be 0, 1, 2 and for the parallel adder. $C_{n-1}$ is the carry from the previous adder.

**Table 2.5 Truth Table of a 4-bit Parallel Adder**

| *Row Number* | $C_{n-1}$ | $A_n$ | $B_n$ | $S_n$ | $C_n$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 2 | 0 | 1 | 0 | 1 | 0 |
| 3 | 0 | 1 | 1 | 0 | 1 |
| 4 | 1 | 0 | 0 | 1 | 0 |
| 5 | 1 | 0 | 1 | 0 | 1 |
| 6 | 1 | 1 | 0 | 0 | 1 |
| 7 | 1 | 1 | 1 | 1 | 1 |

} No carry generation i.e., $C_{out} = 0$

} Carry propagation, i.e., $C_{out} = C_{in}$

} Carry generation, i.e., $C_{out} = 1$

***Ex. With truth table find the sum and output carry for the addition of the following two 4-bit numbers if the input carry ($C_{n-1}$) is 0 –***

$$A_3 A_2 A_1 A_0 = 1100 \text{ and } B_3 B_2 B_1 B_0 = 1100$$

***Sol.*** For n = 0, $A_0 = 0$, $B_0 = 0$ and $C_{n-1} = 0$. With the first row of the table, we get –

$$S_0 = 0 \text{ and } C_0 = 0$$

For n = 1, $A_1 = 0$, $B_1 = 0$ and $C_{n-1} = 0$. With the first row of the table $S_1 = 0$ and $C_1 = 0$.

For n = 2, $A_2 = 1$, $B_2 = 1$ and $C_{n-1} = 0$. With the fourth row of the table, $S_2 = 0$ and $C_2 = 1$.

For n = 3, $A_3 = 1$, $B_3 = 1$ and $C_{n-1} = 1$. With the last row of the truth table, $S_3 = 1$ and $C_3 = 1$. the term $C_4$ becomes the output carry, the sum of $A_3 A_2 A_1 A_0 = 1100$ and $B_3 B_2 B_1 B_0 = 1100$ is $S_3 S_2 S_1 S_0 = 1100$.

***Q.21. Design a full adder using minimum logic gates and also discuss the working of parallel adder.*** ***(R.G.P.V., Dec. 2012)***

***Ans.*** **Full Adder** – Refer the ans. of Q.7.

**Parallel Adder** – Refer the ans. of Q.20.

***Q.22. Explain serial and parallel addition with diagram.*** ***(R.G.P.V., June 2011)***

***Ans.*** **Serial Adder** – Refer the ans. of Q.19.

**Parallel Adder** – Refer the ans. of Q.20.

***Q.23. What is the disadvantage of serial adders ? For which applications are they preferred ?***

***Ans.*** Serial adders are slower than the parallel adders, since they require one clock pulse per pair of bits to be added. Serial adders are used where circuit minimization is more important than speed as in pocket calculators.

***Q.24. Compare serial adder with parallel adder.***

***Ans.*** The comparison between serial and parallel adders are as follows –

| *S.No.* | *Serial Adder* | *Parallel Adder* |
|---|---|---|
| (i) | Serial-adder is less faster. | Generally the parallel-adder is faster. |
| (ii) | It needs less components. | It needs more components. |
| (iii) | The addition is performed bit by bit starting with the least significant bit (LSB). | All the bits are added simultaneously. |

| | | |
|---|---|---|
| (iv) | Serial-adder uses shift registers. | The parallel-adder uses reg with parallel load. |
| (v) | Serial-adder requires only one full-adder circuit and a carry flip-flop. | The number of full-adder c in a parallel-adder is equal t number of bits in the binary num |
| (vi) | The serial-adder is a sequential circuit. | The parallel-adder is a co national circuit. |

***Q.25. Discuss the design of an n-bit ripple carry address which full adders.*** ***(R.G.P.V., Dec.***

***Ans.*** When two n-bit binary numbers are to be added, the number adders needed will be equal to the number of bits n in each number. The out of each full-adder is connected to the carry-in of the next highe adder. The sum and carry-out bits of any stage cannot be produced some time after the carry-in of that stage occurs. This is due to the prop delays in the logic circuitry, which lead to a time delay in the addition p The time between the application of the carry-in and the occurrence carry-out is the carry propagation delay for each full adder.

Fig. 2.24 shows that the sum ($S_0$) and carry-out ($C_{out}$) bits gi $FA_0$ are not valid, until after the propagation delay of $FA_0$. Similarly not valid until after the cumulative propagation delay of two full adde $FA_0$ and $FA_1$, and so on. At each stage, the sum bit is not valid until a carry bits in all the preceding stages are valid. In effect, carry bit propagate or ripple through all stages before the most significant sum valid. Hence, the total sum is not valid until after the cumulative delay the adders.

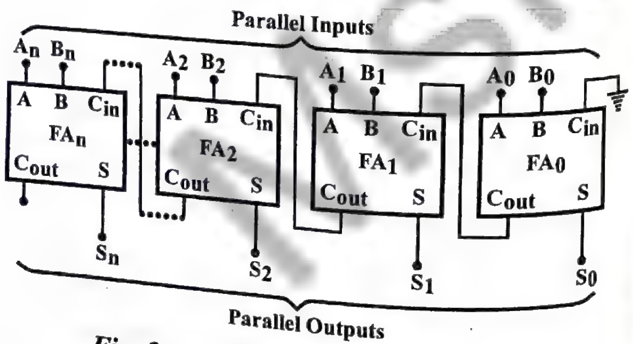


***Fig. 2.24 n-bit Parallel Binary Adder***

The parallel adder, in which the carry-out of each full adder is the in to the next most significant adder is called a ripple carry adder. The the number of bits that a ripple carry adder must add, the greater the required for it to perform a valid addition.

***Q.26. Draw the logic diagram of look-ahead carry generator and explain its working.*** ***(R.G.P.V., Dec. 2008, 2015)***

***Or***

***Explain the operation of look-ahead carry generator.*** ***(R.G.P.V., June 2007, 2008, 2010)***

***Or***

***Discuss/Explain the working of look ahead carry generator.*** ***(R.G.P.V., Dec. 2012, June 2015)***

***Or***

***What is look ahead carry generator ? Explain with logic diagram.*** ***(R.G.P.V., June 2014)***

***Or***

***Design and explain the working of look ahead carry generator.*** ***(R.G.P.V., Dec. 2014)***

***Or***

***Explain look ahead carry generator.*** ***(R.G.P.V., Dec. 2016)***

***Ans.*** To design fast operating parallel adders, we can use gates with lower propagation delay time. The delay time of the adder will increase with increasing number of bits to be added. The look-ahead carry is most commonly used. Although it requires additional circuitry but the speed of the adder becomes independent of the number of bits.

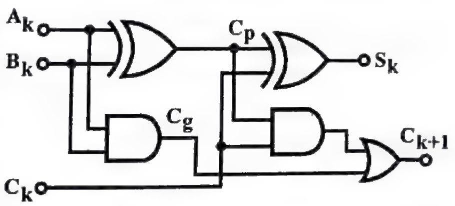


***Fig. 2.25 Implementation of a Full-adder using Logic Gates***

Let us consider a full-adder circuit with EX-OR realization, as shown in fig. 2.25. With this figure, we get

$$C_p = A_k \oplus B_k \quad ...(i)$$
$$C_g = A_k B_k \quad ...(ii)$$
$$S_k = C_p \oplus C_k = A_k \oplus B_k \oplus C_k \quad ...(iii)$$
$$C_{k+1} = C_g + C_p C_k \quad ...(iv)$$

The output $C_g$ of the first half-adder is 1 if $A_k$ and $B_k$ both are 1 and a carry is generated. The variable $C_g$ is known as carry generate. Its value is independent of the input carry. The variable $C_p$ is known as a carry propagate because this is the term associated with the propagation of the carry from $C_k$ to $C_{k+1}$.

In a similar manner; for an n-stage adder, the final carry $C_{n-1}$ can be determined. Here we consider a 4-bit adder and write Boolean expressions for the carry outputs $C_0$, $C_1$, $C_2$ and $C_3$.

$$C_{p0} = A_0 \oplus B_0$$

and

$$C_{g0} = A_0B_0$$

$$C_{in} = C_1 = C_{g0} + C_{p0}\, C_{k0}$$

$$C_2 = C_{g1} + C_{p1}\, C_1$$

$$C_3 = C_{g2} + C_{p2}\, C_{g1} + C_{p2}\, C_{p1}\, C_1$$

$$C_4 = C_{g3} + C_{p3}\, C_{g2} + C_{p3}\, C_{p2}\, C_{g1} + C_{p3}\, C_{p2}\, C_{p1}\, C_1$$

Since the Boolean function for each output carry is expressed in sum of products, each function can be implemented with one level of AND gates followed by an OR gate. The three Boolean functions for $C_2$, $C_3$ and $C_4$ are implemented in the look-ahead carry generator as depicted in fig. 2.26. Here we note that $C_4$ does not have to wait for $C_3$ and $C_2$ to propagate.

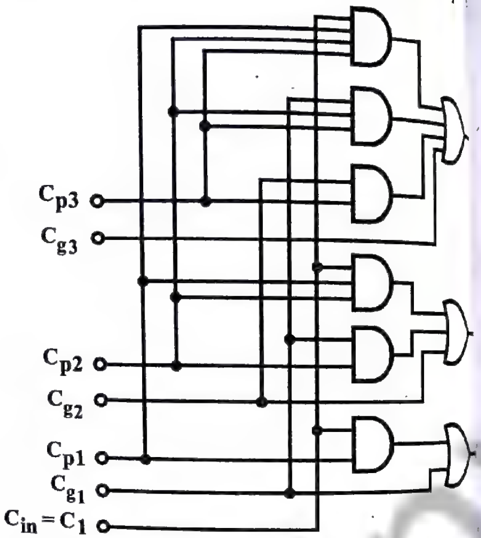


*Fig. 2.26 Logic Diagram of Look-ahead C*
*Generator*

***Q.27. How does the look-ahead carry adder speed up the addition p***

***Ans.*** In case of the parallel adder, the speed with which an additi be performed is governed by the time required for the carries to propa ripple through all of the stages of the adder. The look-ahead carry adder up the process by eliminating this ripple carry delay. It examines all the bits simultaneously and also generates the carry-in bits for all the simultaneously.

***Q.28. When is a carry generated and when is a carry propagated***

***Ans.*** The carry generate (CG) function indicates as to when a ca would be generated by the full-adder. A carry-out is generated only whe the input bits are 1. This condition is expressed as the AND function two input bits A and B. Thus,

$$CG = A.B$$

A carry-in may be propagated by the full-adder when either or both input bits are 1. This condition is expressed as the OR function of the in and B. Thus,

$$CP = A+B$$

***Q.29. Explain/Design BCD adders.***

***(R.G.P.V., June 2008, 2012, Dec. 2013)***

***Or***

***Design and explain the working of 4-bit BCD adder.***

***(R.G.P.V., Feb. 2010)***

***Or***

***Design a BCD adder and also give the rules of BCD addition.***

***(R.G.P.V., June 2010)***

***Or***

***Design a BCD adder using logic gates. (R.G.P.V., Dec. 2012)***

***Or***

***Design and explain the working of BCD adder. (R.G.P.V., Dec. 2014)***

***Or***

***Draw the logic diagram of BCD adder and explain its working.***

***(R.G.P.V., June 2007, Dec. 2015)***

***Or***

***Write short note on BCD adders. (R.G.P.V., Dec. 2017)***

***Ans.*** A BCD adder is a circuit that adds two BCD digits in parallel and produces a sum digit also in BCD. A BCD adder must include the correction logic in its internal logic.

Here we consider the arithmetic addition of two decimal digits in BCD, with a possible carry from previous stage. Since each input digit does not exceeds 9, the output sum cannot be greater than 9 + 9 + 1 = 19, the 1 in the sum being an input carry. Suppose we apply two BCD digits to a 4-bit binary adder. The adder will make the sum in binary and produce a result that may range from 0 to 19. These binary numbers are given in table 2.6 and are labeled by symbols K, $Z_8$, $Z_4$, $Z_2$, $Z_1$. Here K is the output carry and the subscripts under Z represents weights 8, 4, 2 and 1 that can be assigned to the 4-bits in the BCD code.

In table 2.6, it is obvious that when the binary sum is equal to or less than 1001, the corresponding BCD number is identical and therefore no conversion is required. When the binary sum is greater than 1001, we obtain a non-valid BCD code. The addition of binary 6(0110) to the binary sum converts it to the correct BCD representation and also produces an output carry as required.

**Table 2.6**

| | *Binary Sum* | | | | | *BCD Sum* | | | | *Decimal* |
|---|---|---|---|---|---|---|---|---|---|---|
| *K* | $Z_8$ | $Z_4$ | $Z_2$ | $Z_1$ | $C_{out}$ | $S_8$ | $S_4$ | $S_2$ | $S_1$ | *Number* |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 2 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 3 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 4 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 5 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 6 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 7 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 8 |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 9 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 10 |
| 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 11 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 12 |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 13 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 14 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 15 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 16 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 17 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 18 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 19 |

The correction is required when the binary sum has an output carry K 1. The other six combination from 1010 to 1111 that need a correction hav 1 in position $Z_8$. To distinguish them from binary 1000 and 1001, which have a 1 in position $Z_8$. We specify further that either $Z_4$ or $Z_2$ must have The condition for a correction and an output carry can be expressed by Boolean function as–

$$C = K + Z_8Z_4 + Z_8Z_2$$

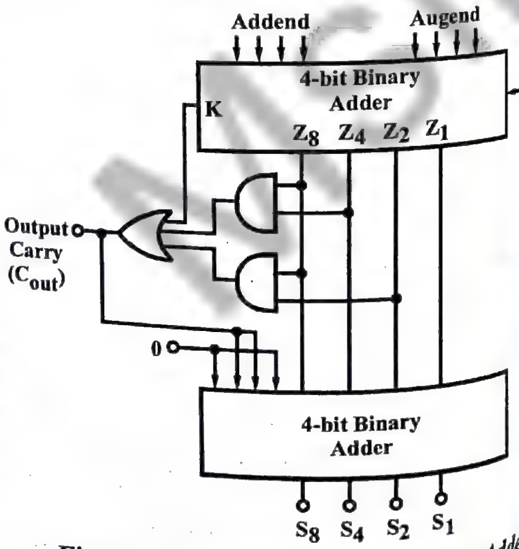


Fig. 2.27 Block Diagram of a BCD Adder

When C = 1, it is necessary to add 0110 to the binary sum and provide an output carry for the next stage.

To add 0110 to the binary adder, we use a two 4-bit binary adder as shown in fig. 2.27.

The two decimal digits, together with the input carry are first added in the top 4-bit binary adder to produce the binary sum. When the output carry equal to zero, nothing is added to the binary sum. When it is equal to 1, binary 0110 is added to the binary sum through the bottom 4-bit binary adder. The output carry generated from the bottom binary adder can be ignored, since it supplies information already available at the output-carry terminal.

***Q.30. Draw and explain a 4-bit magnitude comparator.***

***(R.G.P.V., June 2014)***

***Ans.*** A 4-bit magnitude comparator compares two 4-bit numbers A and B and provides one of the following outputs, A = B, A < B and A > B. Let $A = A_3A_2A_1A_0$ and $B = B_3B_2B_1B_0$ be the two 4-bit numbers to be compared. The steps involved in comparing two 4-bit numbers are –

(i) Examine the two most significant bits ($A_3$ and $B_3$). If $A_3 > B_3$, then A > B. If $A_3 < B_3$, then A < B. If $A_3 = B_3$, no conclusion can be drawn regarding the relative magnitudes of the two numbers and the next pair of bits ($A_2$ and $B_2$) must be examined.

(ii) If $A_3 = B_3$ and $A_2 > B_2$, then A > B. If $A_3 = B_3$ and $A_2 < B_2$, then A < B. Although, if $A_3 = B_3$ and $A_2 = B_2$, no decision can be made regarding the relative magnitudes of the two numbers and the next pair of bits ($A_1$ and $B_1$) must be examined.

(iii) If $A_3 = B_3, A_2 = B_2$ and $A_1 > B_1$ then A > B. If $A_3 = B_3, A_2 = B_2$ and $A_1 < B_1$, then A < B. Although, if $A_3 = B_3$, $A_2 = B_2$ and $A_1 = B_1$, no conclusion can yet be made regarding the relative magnitudes of the two numbers and the LSBs ($A_0$ and $B_0$) must be examined.

(iv) If $A_3 = B_3, A_2 = B_2, A_1 = B_1$ and $A_0 > B_0$, then A > B. If $A_3 = B_3$, $A_2 = B_2, A_1 = B_1$ and $A_0 < B_0$, then A < B. However, if $A_3 = B_3, A_2 = B_2, A_1 = B_1$ and $A_0 = B_0$, then A = B.

If the most significant bits are equal (i.e., $A_3 = B_3 = 0$ OR $A_3 = B_3 = 1$), then

$$E_3 = \overline{A}_3\overline{B}_3 + A_3B_3 = \overline{A_3 \oplus B_3}$$

If the next two most significant bits are equal, then

$$E_2 = \overline{A}_2\overline{B}_2 + A_2B_2 = \overline{A_2 \oplus B_2}$$

If the next two most significant bits are equal, then

$$E_1 = \overline{A}_1\overline{B}_1 + A_1B_1 = \overline{A_1 \oplus B_1}$$

If the two least significant bits are equal, then

$$E_0 = \overline{A}_0\overline{B}_0 + A_0B_0 = \overline{A_0 \oplus B_0}$$

Therefore, if A = B, then

$$E = E_3E_2E_1E_0$$

$$= (\overline{A_3 \oplus B_3})\,(\overline{A_2 \oplus B_2}).(\overline{A_1 \oplus B_1})\,(\overline{A_0 \oplus B_0}) = 1$$

The expression for determining whether A > B is

$$A > B = A_3\overline{B}_3 + E_3A_2\overline{B}_2 + E_3E_2A_1\overline{B}_1 + E_3E_2E_1A_0\overline{B}_0$$

The first term in this equation $A_3\overline{B}_3 = 1$ if $A_3 > B_3$ and, if that is the case, then A > B. The second term in this equation $E_3A_2\overline{B}_2 = 1$ if $A_3 = B_3$ and $A_2 > B_2$ and, if that is the case, then A > B. The third term $E_3E_2A_1\overline{B} = 1$ if $A_3 = B_3$, $A_2 = B_2$ and $A_1 > B_1$. If these three conditions are satisfied, then A > B. Finally, the fourth term $E_3E_2E_1A_0\overline{B}_0 = 1$ IF $A_3 = B_3, A_2 = B_2, A_1 = B_1$ and $A_0 = B_0$. If these four conditions are satisfied, then A > B.

The expression for determining whether A < B is

$$A<B = \overline{A}_3B_3 + E_3\overline{A}_2B_2 + E_3E_2\overline{A}_1B_1 + E_3E_2E_1\overline{A}_0B_0$$

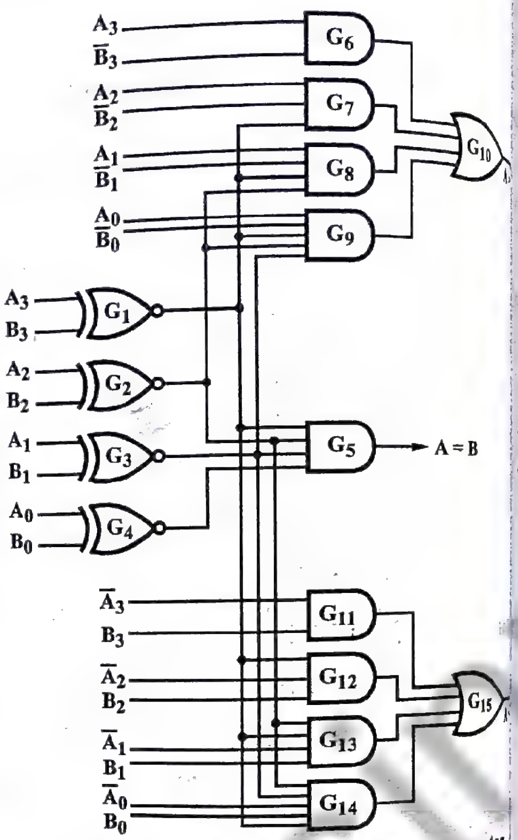


Fig. 2.28 4-bit Magnitude Comparator

This has the same form as the A > B expression and can be analysed similar way.

The implementation of a 4-bit magnitude comparator using EX-NOR AND gates using the above expression is shown in fig. 2.28.

## MULTIPLEXER-DEMULTIPLEXER

***Q.31. Write a short note on multiplexer.*** *(R.G.P.V., May*

***Or***

***Explain multiplexer circuit.*** *(R.G.P.V., Dec.*

***Ans.*** A multiplexer is a logic circuit that gets several inputs to a s output. The selection of a particular input line is controlled by a set of sele lines. The block diagram of a multiplexer with n input lines and one output line as illustrated in fig. 2.29. For selecting one output of n inputs for connection to the output, a set of m select inputs are required, where $2^m = n$. Depending upon the digital code applied at the select inputs on out of n data sources is selected and transmitted to a single output channel. Generally a strobe input (G) is incorporated, which helps in cascading and it is active low, which means that it performs its intended operation when it is low (or 0). For example, to select 1 out of 4 input lines, two select lines are required, to select 1 of 8 input lines, three select lines are required and so on.

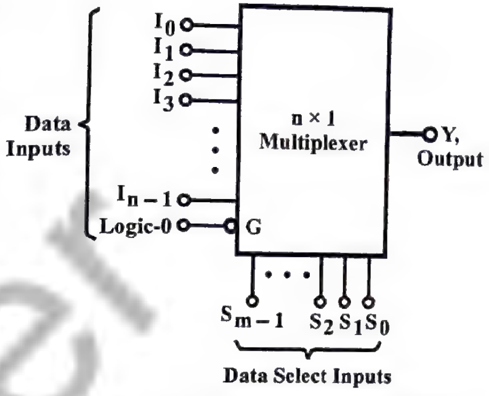
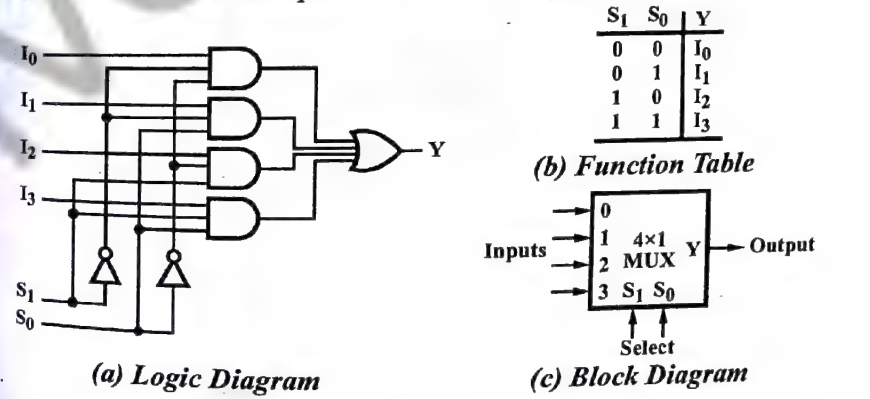


Fig. 2.29 Block Diagram of a Digital Multiplexer

***Q.32. Draw and explain 4 × 1 multiplexer.*** *(R.G.P.V., Dec. 2017)*

***Ans.*** A 4 to 1 line multiplexer is shown in fig. 2.30.



| $S_1$ | $S_0$ | Y |
|---|---|---|
| 0 | 0 | $I_0$ |
| 0 | 1 | $I_1$ |
| 1 | 0 | $I_2$ |
| 1 | 1 | $I_3$ |

(a) Logic Diagram

(b) Function Table

(c) Block Diagram

Fig. 2.30 4 to 1 Line Multiplexer

Each of the four input lines, $I_0$ to $I_3$ is applied to one input of an AND gate. Selection lines $S_1$ and $S_0$ are decoded to select a particular AND gate. To demonstrate the circuit operation consider the case when $S_1S_0 = 10$. The AND gate associated with input $I_2$ has two of its inputs equal to 1 and the third input connected to $I_2$. The other three AND gates have at least one input equal to 0, which makes their output equal to 0. The OR-gate output is now equal to the value of $I_2$, thus providing a path from the selected input to the output. A multiplexer is also called a data selector, since it selects one of many inputs and steers the binary information to the output line.

**Q.33. Explain the working of multiplexer and draw the detailed of 4 to 1 line multiplexer.** **(R.G.P.V., June**

*Ans.* Refer the ans. of Q.31 and Q.32.

**Q.34. Explain the working of 8 to 1 (or 8 × 1) multiplexer.** **[R.G.P.V., Dec. 2002**

*Ans.* The logic diagram of 8 to 1 multiplexer is illustrated in fig. This multiplexer has eight data inputs ($D_0 - D_7$), three data select input ($S_2 - S_0$) and an enable input ($\overline{EN}$). A low on the $\overline{EN}$ input allow selected input data to pass through to the output. When $\overline{EN}$ is high (or multiplexer is disabled, so that output Y = 0, regardless of the select code. Here we note that the output as well as its complement are ava The working of this multiplexer is given in the truth table of table 2.7.

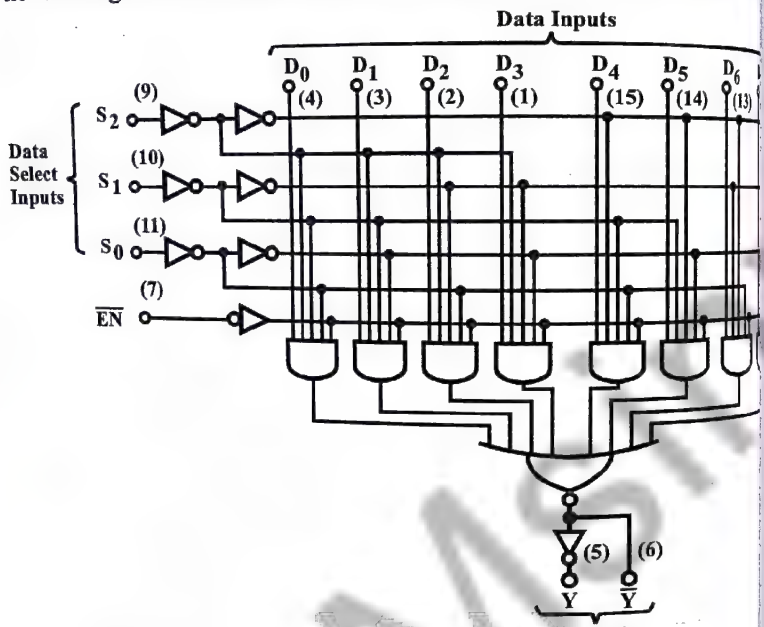


**Fig. 2.31 Logic Diagram of 8 to 1 (or 8 × 1) Multiplexer**

**Table 2.7 Truth Table of 8 to 1 (or 8 × 1) Multiplexer**

| *Enable Signal* | *Inputs* | | | *Outputs* | |
|---|---|---|---|---|---|
| $(\overline{EN})$ | $S_2$ | $S_1$ | $S_0$ | $Y$ | $\overline{Y}$ |
| 1 | × | × | × | 0 | 1 |
| 0 | 0 | 0 | 0 | $D_0$ | $\overline{D_0}$ |
| 0 | 0 | 0 | 1 | $D_1$ | $\overline{D_1}$ |
| 0 | 0 | 1 | 0 | $D_2$ | $\overline{D_2}$ |
| 0 | 0 | 1 | 1 | $D_3$ | $\overline{D_3}$ |
| 0 | 1 | 0 | 0 | $D_4$ | $\overline{D_4}$ |
| 0 | 1 | 0 | 1 | $D_5$ | $\overline{D_5}$ |
| 0 | 1 | 1 | 0 | $D_6$ | $\overline{D_6}$ |
| 0 | 1 | 1 | 1 | $D_7$ | $\overline{D_7}$ |

**Q.35. Implement a full-adder circuit with multiplexers.** **(R.G.P.V., Nov./Dec. 2007, June 2017)**

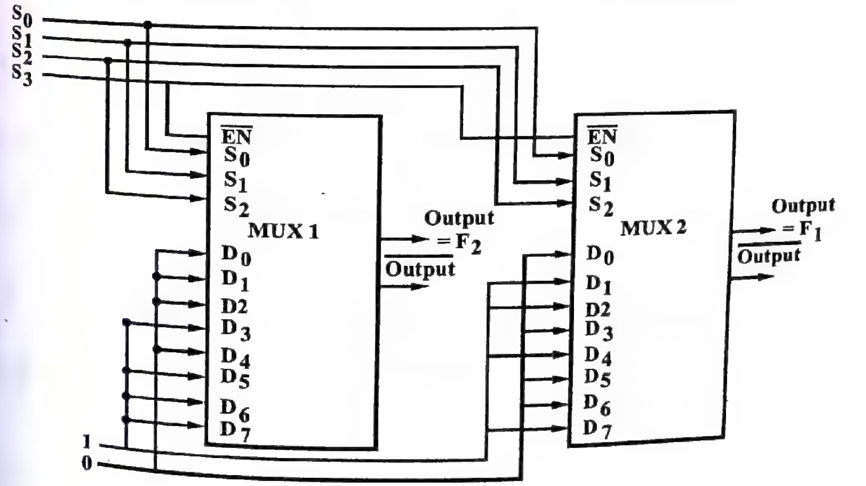
Ans. The truth tables for the logic functions $F_1 = A \oplus B \oplus C_{in}$, $F_2 = (A \oplus B) C_{in} + AB$ and the logic diagram to implement $F_1$ and $F_2$ are shown in fig. 2.32. Since, there are three input variables, we can use a multiplexer with three data select inputs (i.e., an 8-to-1 MUX). The truth tables in fig. 2.32 show the use of data select inputs $S_2$ , $S_1$, and $S_0$ for input variables A, B, and $C_{in}$ respectively, since $F_1 = 1$ when $ABC_{in}$ = 001, 010, 100 and 111,

| $S_2$<br>A | $S_1$<br>B | $S_0$<br>$C_{in}$ | $F_1 = A \oplus B \oplus C_{in}$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

*(a) Truth Table*

| $S_2$<br>A | $S_1$<br>B | $S_0$<br>$C_{in}$ | $F_2 = (A \oplus B)C_{in} + AB$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

*(b) Truth Table*



*(c) Logic Diagram*

**Fig. 2.32 Logic Diagram of Full-adder using Multiplexer**

we connect logical 1 to data inputs $D_1$, $D_2$, $D_4$ and $D_7$. Logical 0 is conn to other data inputs $D_0$, $D_3$, $D_5$ and $D_6$. When the data select inputs ar of the combinations for which $F_1 = 1$, the output will be 1, and whe data select inputs are any of the combinations for which $F_1 = 0$, the o will be 0.

Similarly, $F_2 = 1$ when $ABC_{in}$ = 011, 101, 110 and 111, we co logical 1 to data inputs $D_3$, $D_5$, $D_6$ and $D_7$. Logical 0 is connected to data inputs $D_0$, $D_1$, $D_2$ and $D_4$. When the data select inputs are any c combinations for which $F_2 = 1$, the output will be 1, and when the data s inputs are any of the combinations for which $F_2 = 0$, the output will be

***Q.36. Write short note on demultiplexer.*** *(R.G.P.V., Nov.*

***Ans.*** A demultiplexer is a logic circuit that receives information on a s line and transmits this information on one of ($2^m$) several outputs. The sele of a specific output line is controlled by the bit values of m select lines number of output lines is n and the number of select lines is m, (i.e., n = $2^m$). This circuit can also be used as binary to decimal decoder with binary inputs applied at the select input lines and the output will be obtained on the corresponding line. The data input line is to be connected to logic-1 level. The block diagram of demultiplexer is shown in fig. 2.33.

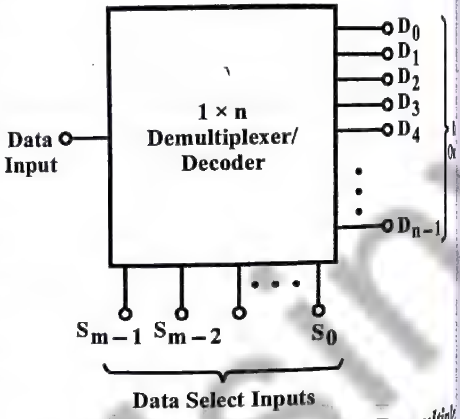


*Fig. 2.33 Block Diagram of Demultiple*

***Q.37. Explain multiplexers and demultiplexers.***
*(R.G.P.V., June 2007, Dec. 2008,*
***Or***
***Discuss the multiplexer and demultiplexer.*** *(R.G.P.V., June*

***Ans.*** Refer the ans. of Q.31 and Q.36.

***Q.38. What are the applications of multiplexer and demultiplexe***

***Ans.* Applications of Multiplexer –**

(i) The multiplexers can be used to route data from one of se sources to one destination.

(ii) A multiplexer can also be used as a control sequencer.

(iii) It can be used to implement logic functions in sum of pr (SOP) form with the truth table directly.

(iv) A multiplexer can be used for waveform generation.

(v) The data in parallel form is converted to serial form using multiplexer.

**Applications of Demultiplexer –**

(i) Demultiplexers are used as clock demultiplexers in synchronous data transmission systems in the receivers and in security monitoring systems etc.

(ii) This device is very useful if multiple output combinational circuit is to be designed, because this requires minimum package count.

***Q.39. Draw and explain the logic diagram of 1 to 4 (or 1 × 4) demultiplexer.***

***Ans.*** The demultiplexer is also known as a serial to parallel converter. For example, consider 1 to 4 demultiplexer, it has a single data input (D), four outputs ($Y_0$ to $Y_3$) and two data select inputs ($S_0$ and $S_1$). The truth table of the 1 to 4 demultiplexer is given in table 2.8.

**Table 2.8 Truth Table of 1 to 4 (or 1 × 4) Demultiplexer**

| *Data Input* | *Data Select Inputs* | | *Outputs* | | | |
|---|---|---|---|---|---|---|
| ***D*** | $S_1$ | $S_0$ | $Y_3$ | $Y_2$ | $Y_1$ | $Y_0$ |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 |

From the truth table, we noted that the data input is connected to the output $Y_0$ when the data select inputs $\overline{S}_1\overline{S}_0$ = 00, the data input is connected to the output $Y_1$ when the $\overline{S}_1 S_0$ = 01, the data input is connected to the outputs $Y_2$ and $Y_3$ when the $S_1\overline{S}_0$ = 10 and $S_1S_0$ = 11, respectively. The expression for the outputs can be written as –

$$Y_0 = \overline{S}_1\overline{S}_0 D \quad ...(i)$$

$$Y_1 = \overline{S}_1 S_0 D \quad ...(ii)$$

$$Y_2 = S_1\overline{S}_0 D \quad ...(iii)$$

$$Y_3 = S_1 S_0 D \quad ...(iv)$$

Using the above expressions, a 1 to 4 (or 1 × 4) demultiplexer can be implemented, as shown in fig. 2.34.

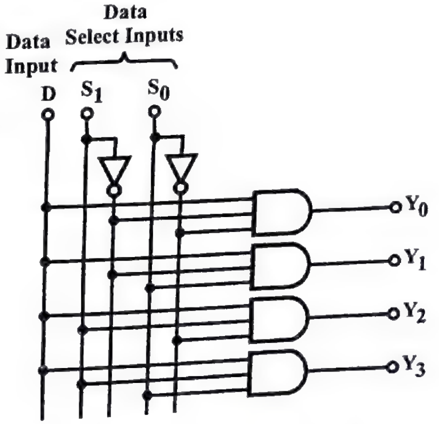


*Fig. 2.34 Logic Diagram of 1 to 4 (or 1 × 4) Demultiplexer*

## NUMERICAL PROBLEMS

***Prob.1. Implement the function –***

***$F(A, B, C, D) = \Sigma (0, 1, 3, 4, 7, 8, 10, 13)$ using multiplexer.***

***(R.G.P.V., June 201***

***Sol.*** As the function is a four variables function, a multiplexer wit select lines and eight input is required. Apply variables B, C and D to t select lines. The procedure for implementing the function is shown table 2.9.

**Table 2.9 Implementation Table**

| | $D_0$ | $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_5$ | $D_6$ | $D_7$ |
|---|---|---|---|---|---|---|---|---|
| $\bar{A}$ | ⓪ | ① | 2 | ③ | ④ | 5 | 6 | ⑦ |
| A | ⑧ | 9 | ⑩ | 11 | 12 | ⑬ | 14 | 15 |
| Y | 1 | $\bar{A}$ | A | $\bar{A}$ | $\bar{A}$ | A | 0 | $\bar{A}$ |

In the truth table in table 2.9, the first half of the minterms are associa with $\bar{A}$ and the second half with A. Now, using the table, the given functi can be implemented using 8 to 1 multiplexer as shown in fig. 2.35.

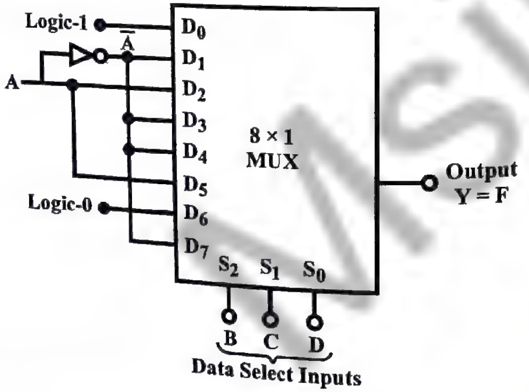


*Fig. 2.35*

***Prob.2. Implement the function $F(A, B, C, D) = \Sigma(0, 1, 3, 4, 8, 9,$ with a multiplexer.***

***(R.G.P.V., June 20***

***Sol.*** This is a four-variable function and therefore we need a multiple with three selection lines and eight inputs. We select to apply variables B, C D to the selection lines. The implementation table is given in table 2.10.

The first half of the minterms are associated with $\bar{A}$ and the second half with A. By circling the minterms of the given function and applying the rules for finding values for the multiplexer inputs we obtain the implementation as shown in fig. 2.36.

**Table 2.10 Implementation Table**

| | $I_0$ | $I_1$ | $I_2$ | $I_3$ | $I_4$ | $I_5$ | $I_6$ | $I_7$ |
|---|---|---|---|---|---|---|---|---|
| $\bar{A}$ | ⓪ | ① | 2 | ③ | ④ | 5 | 6 | 7 |
| A | ⑧ | ⑨ | 10 | 11 | 12 | 13 | 14 | ⑮ |
| | 1 | 1 | 0 | $\bar{A}$ | $\bar{A}$ | 0 | 0 | A |

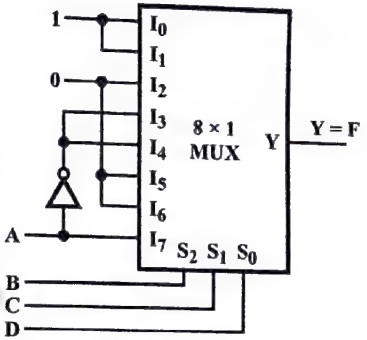


***Fig. 2.36 Implementing $F(A, B, C, D) = \Sigma (0, 1, 3, 4, 8, 9, 15)$ using 8 × 1 MUX***

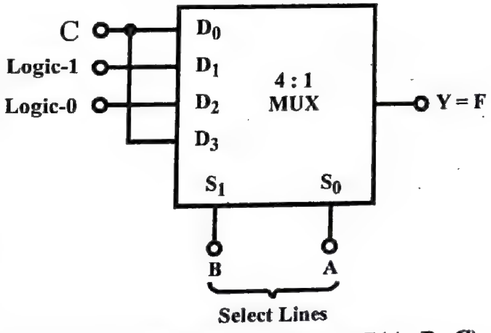
***Prob.3. Implement the following Boolean function using 4:1 multiplexer using A and B variables to the selection lines –***

***$F(A, B, C) = \Sigma m(1, 4, 5, 7)$. (R.G.P.V., Dec. 2015)***

***Sol.*** The given function has three variables. Therefore, it can be implemented using a multiplexer with two select lines and four data inputs. The implementation table of the given function is shown in table 2.11.

**Table 2.11 Implementation Table**

| | $D_0$ | $D_1$ | $D_2$ | $D_3$ |
|---|---|---|---|---|
| $\bar{C}$ | 0 | ① | 2 | 3 |
| C | ④ | ⑤ | 6 | ⑦ |
| Y | C | 1 | 0 | C |



Now, the given function can be implemented using 4 : 1 multiplexer as shown in fig. 2.37.

***Fig. 2.37 Implementation of $F(A, B, C) = \Sigma m (1, 4, 5, 7)$ using 4 : 1 Multiplexer***

***Prob.4. Realize the following function using multiplexer –***

***$\Sigma(0, 3, 5, 7, 11, 14)$.***

***(R.G.P.V., Dec. 2012)***

***Sol.*** As the function is a four-variable function, a multiplexer with 3 select lines and eight inputs is required. Apply variables B, C and D to the select lines. The procedure for implementing the function is shown in table 2.12.

**Table 2.12 Procedure for Implementation of the Function**

| | $D_0$ | $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_5$ | $D_6$ | $D_7$ |
|---|---|---|---|---|---|---|---|---|
| $\overline{A}$ | ⓪ | 1 | 2 | ③ | 4 | ⑤ | 6 | ⑦ |
| A | 8 | 9 | 10 | ⑪ | 12 | 13 | ⑭ | 15 |
| Y | $\overline{A}$ | 0 | 0 | 1 | 0 | $\overline{A}$ | A | $\overline{A}$ |

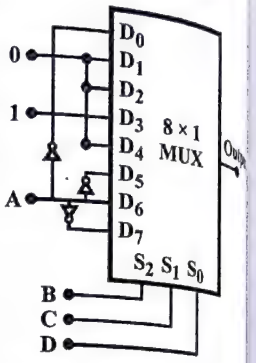


*Fig. 2.38*

Now, using the table, the given function can be implemented using an 8-to-1 multiplexer as shown in fig. 2.38.

***Prob.5. Implement the following using multiplexer –***

***Σ (4, 5, 7, 9, 12, 15)*** ***(R.G.P.V., June 201***

***Sol.*** As the function is a four-variables function, a multiplexer with 3 sele lines and eight inputs is required. Apply variables B, C and D to the select li The procedure for implementing the function is shown in table 2.13.

**Table 2.13 Procedure for Implementation of the Function**

| | $D_0$ | $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_5$ | $D_6$ | $D_7$ |
|---|---|---|---|---|---|---|---|---|
| $\overline{A}$ | 0 | 1 | 2 | 3 | ④ | ⑤ | 6 | ⑦ |
| A | 8 | ⑨ | 10 | 11 | ⑫ | 13 | 14 | ⑮ |
| Y | 0 | A | 0 | 0 | 1 | $\overline{A}$ | 0 | 1 |

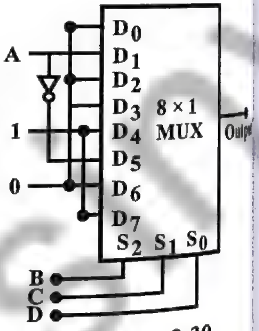


*Fig. 2.39*

Now, using the table, the given function can be implemented using an 8-to-1 multiplexer as shown in fig. 2.39.

## ENCODER-DECODER, ARITHMETIC CIRCUITS, ALU

***Q.40. What is decoder ? Explain how combinational circuits like f adder can be implemented with decoder.*** ***(R.G.P.V., May 201***

***Ans.*** A decoder is a logic circuit that converts an N-bit binary input co into M output lines such that only one output line is activated for each one the possible combinations of inputs. The general decoder diagram with inputs and M outputs is shown in fig. 2.40. Because each of the N inputs c be a 0 or 1, there are $2^N$ possible input combinations or codes. For each these input combinations, only one of the M outputs will be active (High). other outputs will remain inactive (Low). Some decoders are designed to produce active low output, while all the other outputs remain high.

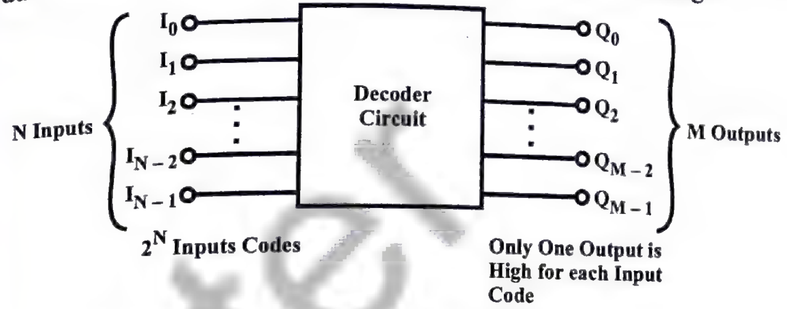


***Fig. 2.40 Block Diagram of a Decoder***

In decoder, the number of outputs is greater than the number of inputs.

**Full Adder with Decoder and OR Gate** – Refer the ans. of Q.11.

***Q.41. What are the important applications of decoders ?***

***Ans.*** Various applications of decoders are as follows –

(i) The decoders are used whenever an output or a group of outputs is to be activated only on the occurrence of a specific combination of input levels.

(ii) They are used in analog to digital converters.

(iii) When the decoder inputs come from a counter that is being continually pulsed, the decoder outputs will be activated sequentially and they can be used as timing or sequencing signals to turn devices 'ON' or 'OFF' at specific times.

(iv) Decoder outputs can be used to drive a display system.

(v) They are widely used in memory systems of computers, where they respond to the address code input from the central processor to activate the memory storage location specified by the address code.

***Q.42. Design a 3 to 8 line decoder.*** ***(R.G.P.V., June 2010)***

***Ans.*** In 3 to 8 line decoder three inputs are decoded into eight outputs, each output representing one of the minterms of the 3 input variables. The three inverters provide the complement of the inputs and each one of the eight AND gates generates one of the minterms.

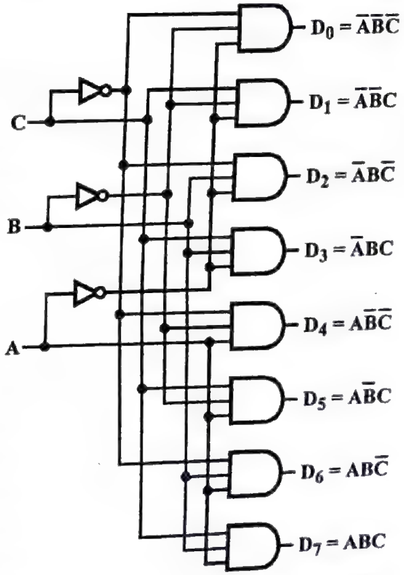

C
B
A
$D_0 = \overline{A}\overline{B}\overline{C}$
$D_1 = \overline{A}\overline{B}C$
$D_2 = \overline{A}B\overline{C}$
$D_3 = \overline{A}BC$
$D_4 = A\overline{B}\overline{C}$
$D_5 = A\overline{B}C$
$D_6 = AB\overline{C}$
$D_7 = ABC$

***Fig. 2.41 3 to 8 Line Decoder***

The operation of the decoder may be further clarified from its input/outp relationship as shown in table 2.14.

Table 2.14 Truth Table

| *Inputs* | *Outputs* | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| *A B C* | *$D_0$* | *$D_1$* | *$D_2$* | *$D_3$* | *$D_4$* | *$D_5$* | *$D_6$* | *$D_7$* |
| 0 0 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 0 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 1 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 1 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 0 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 0 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 1 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 1 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

***Q.43. Explain the working of encoders.*** ***(R.G.P.V., Dec. 201***

***Or***

***Write short note on encoder.*** ***(R.G.P.V., June 2015, 2017, May 201***

***Ans.*** An encoder is a digital circuit that performs the inverse operation of decoder. An encoder is a combinational logic circuit that converts an active inpu signal into a coded output signal. An encoder used to encode (convert) a calculat key-stroke into a binary code that can be processed by the calculator circuits.

The encoders can also be devised to encode various symbols and alphabeti characters. The process of converting from familiar symbols (or number) t a coded format is called encoding. The basic block diagram of encoder wit n-inputs and m-outputs is shown in fig. 2.42. An encoder has n-input lines only one of which is activated at a given time and produces an m-bit outpu code depending on which input is activated.

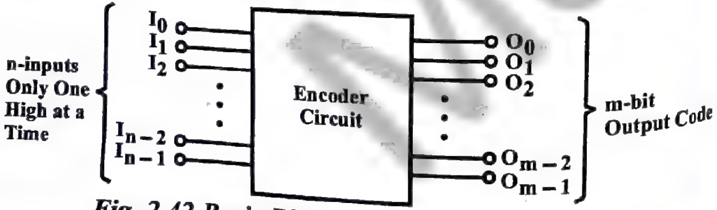


***Fig. 2.42 Basic Block Diagram of an Encoder***

Also refer the ans. of Q.46.

***Q.44. Write short note on decoder and encoder.*** ***(R.G.P.V., June 200***

***Or***

***Explain encoder and decoder.*** ***(R.G.P.V., June 201***

***Ans.*** Refer the ans. of Q.40 and Q.43.

***Q.45. Differentiate between multiplexer and encoder.(R.G.P.V., Dec. 2014)***

***Ans.*** Multiplexer has several data input lines and a single output line. The selection of a particular input line is controlled by a set of selection lines.

An encoder is a combinational logic circuit that converts an active input signal into a coded output signal. In a encoder the number of outputs is less than the number of inputs.

***Q.46. Draw the logic diagram and explain 8-line to 3-line (or octal to binary) encoder.***

***Ans.*** An 8-line to 3-line encoder accepts 8 input lines and produces a 3-bit output code corresponding to the activated input. The truth table of octal to binary encoder is given in table 2.15.

Table 2.15 Truth Table of Octal to Binary Encoder

| *Octal Inputs* | | | | | | | | *Binary Outputs* | | |
|---|---|---|---|---|---|---|---|---|---|---|
| *$D_0$* | *$D_1$* | *$D_2$* | *$D_3$* | *$D_4$* | *$D_5$* | *$D_6$* | *$D_7$* | *$Y_2$* | *$Y_1$* | *$Y_0$* |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

The encoder can be implemented with OR gates whose inputs are determined directly with the truth table. The output $Y_0$ is equal to 1 when the input octal digit $D_1$ or $D_3$ or $D_5$ or $D_7$ is a 1. The output $Y_1$ is 1 for the octal digit $D_2$ or $D_3$ or $D_6$ or $D_7$ and output $Y_2$ are 1 for octal digit $D_4$ or $D_5$ or $D_6$ or $D_7$. These conditions can be expressed by the Boolean functions as follows –

$$Y_0 = D_1 + D_3 + D_5 + D_7$$
$$Y_1 = D_2 + D_3 + D_6 + D_7$$
$$Y_2 = D_4 + D_5 + D_6 + D_7$$

Here we observe that the digit $D_0$ is not present in any of the expressions. So, digit $D_0$ is a don't care. The logic diagram of octal to binary encoder is shown in fig. 2.43.

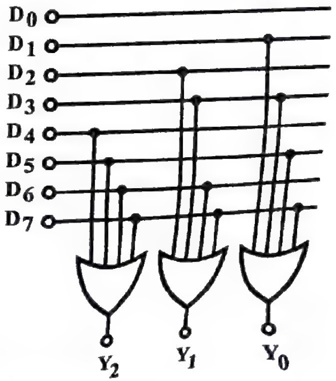


***Fig. 2.43 Logic Diagram of Octal to Binary Encoder***

The encoder in table 2.15 has the disadvantage that only one input can active at any given 'ime. When two inputs are active simultaneously, the out produces an undefined combination.

*Q.47. Explain the 10-line to 4-line encoder.*

*Or*

*Discuss in brief about decimal to BCD encoder. Also draw its logic diag*

*Ans.* The decimal to BCD encoder has ten inputs – one for each deci digit and four outputs corresponding to the BCD code as illustrated in f 2.44. The truth table of decimal to BCD encoder is given in table 2.16.

**Table 2.16 Truth Table of Decimal to BCD Encoder**

| *Decimal Inputs* | | | | | | | | | | *BCD Outputs* | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $D_0$ | $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_5$ | $D_6$ | $D_7$ | $D_8$ | $D_9$ | $Y_3$ | $Y_2$ | $Y_1$ | $Y_0$ |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |

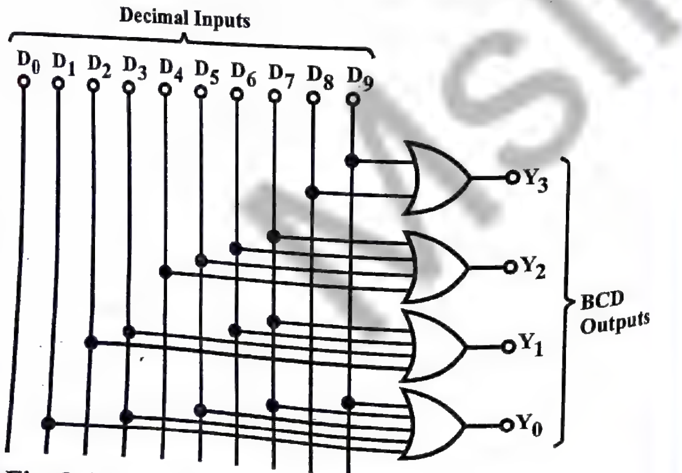


**Fig. 2.44 Logic Diagram of Decimal to BCD Encoder**

There is no explicit input a decimal 0 (or $D_0$). The BCD output is 000 when the decimal inputs 1 to 9 are all 0. From the truth table in table 2.16, w obtain the minimized Boolean expressions as –

$$Y_3 = D_8 + D_9$$
$$Y_2 = D_4 + D_5 + D_6 + D_7$$
$$Y_1 = D_2 + D_3 + D_6 + D_7$$
$$Y_0 = D_1 + D_3 + D_5 + D_7 + D_9$$

*Q.48. Draw the logic diagram and truth table of 4-inputs priority encoder. Explain it.*

*Or*

*Draw the logic diagram of priority encoder and explain its working.*

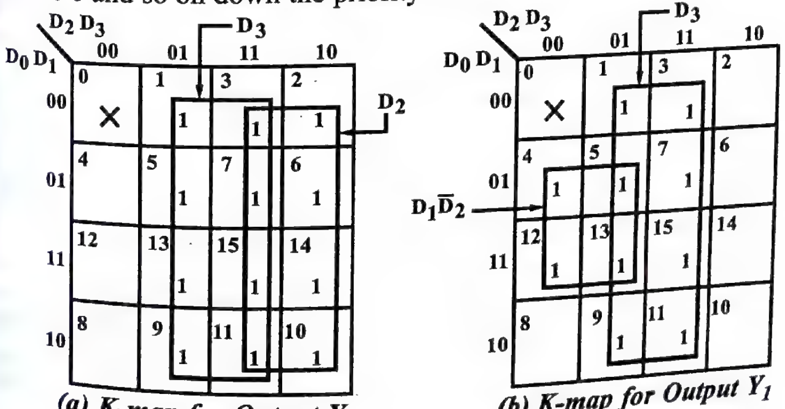
*(R.G.P.V., Dec. 2015)*

*Ans.* A priority encoder is a logic circuit that responds to just one input in accordance with some priority system, between all those which may be simultaneously high (or 1). The operation of the priority encoder is such that when two or more inputs are equal to 1 at the same time, the input having the highest priority will take precedence. The truth table of a four-inputs priority encoder is shown in table 2.17.

**Table 2.17 Truth Table of Four Inputs Priority Encoder**

| *Inputs* | | | | *Ourputs* | | |
|---|---|---|---|---|---|---|
| $D_0$ | $D_1$ | $D_2$ | $D_3$ | $Y_2$ | $Y_1$ | $V$ |
| 0 | 0 | 0 | 0 | × | × | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| × | 1 | 0 | 0 | 0 | 1 | 1 |
| × | × | 1 | 0 | 1 | 0 | 1 |
| × | × | × | 1 | 1 | 1 | 1 |

The ×'s are don't care conditions that designate the fact that the binary value may be equal either 0 or 1. Input $D_3$ has the highest priority, so regardless of the values of the other inputs, if this input is 1, the output for $Y_2$ $Y_1$ is 11. Input $D_2$ has the next priority level. The output is 10 when input $D_2$ is 1 provided that input $D_3$ is 0, regardless of the values of the other two lower priority inputs. The output for $D_1$ input is generated only when higher-priority inputs are 0 and so on down the priority level.



**(a) K-map for Output $Y_2$**

**(b) K-map for Output $Y_1$**

**Fig. 2.45 K-maps for Four Inputs Priority Encoder**

A valid output indicator represented by V, is set to 1 only when more of the inputs are equal to 1. When all inputs are 0, V = 0 and the two outputs of the circuit are not used. The K-map for the minimized Y$_2$ and Y$_1$ is illustrated in fig. 2.45.

The minterms for the two functions are obtained from table 2.17. Al the table has only five rows, when each don't care condition is first re by 0 and then by 1, we obtain all 16 possible input combinations. For example, the third row in the table with × 100 represents minterms 0100 and 1100 since '×' can be assigned either 0 or 1. The minimized Boolean expressions for the four inputs priority encoder are obtained from the K-maps in fig. 2.45. The condition for valid output indicator (V) is an OR function of all the input variables.

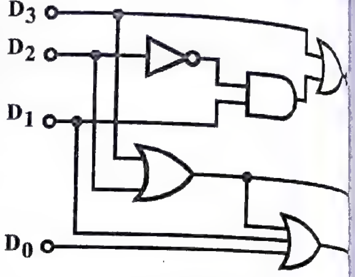


*Fig. 2.46 Logic Diagram of F Inputs Priority Encoder*

$$Y_2 = D_2 + D_3$$

$$Y_1 = D_3 + D_1\overline{D}_2$$

$$V = D_0 + D_1 + D_2 + D_3$$

According to the above Boolean expressions the four inputs pri encoder is drawn in fig. 2.46.

***Q.49. Discuss the arithmetic and logic unit (ALU).***

***Ans.*** The arithmetic logic unit (ALU) of a computer system is the p where the actual execution of the instructions takes place during the proces operation. That is, it is also responsible for all calculations performed an comparisons (decisions) made in the ALU. The data and instructions, st in the primary storage prior to processing, are transferred as and when nee to the ALU where processing takes place. Intermediate results generated i ALU are temporarily transferred back to the primary storage until needed later time. Data may move from memory to ALU and back again to st many times before the processing is over.

There are number of arithmetic and logic operations that a computer perform. These are add, subtract, multiply, divide and logic operation comparisons such as less than, equal to or greater than.

Fig. 2.47 shows, in general terms, how the ALU is interconnected with the rest of the processor. Data are presented to the ALU in registers and the results of an operation are stored in registers. These registers

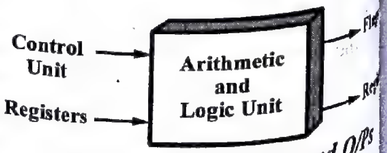


*Fig. 2.47 ALU I/Ps and O/Ps*

are temporary storage locations within the processor that are connected by signal paths to the ALU. The ALU can also set flags as the result of an operation. The flag values are also stored in registers within the processor. The control unit provides signals that control the operation of the ALU and the movement of the data into and out of the ALU.

***Q.50. Write down the functions performed by ALU.***

***Ans.*** ALU (arithmetic and logic unit) is the part of the CPU which performs arithmetic and logic operations. Generally, an ALU performs the following arithmetic and logic operations –

(i) Addition
(ii) Subtraction
(iii) Multiplication
(iv) Division
(v) Logical AND
(vi) Logical OR
(vii) Logical exclusive-OR
(viii) Complement (logical NOT)
(ix) Increment (addition of 1)
(x) Decrement (subtraction of 1)
(xi) Left or right shift (the content of the accumulator can be shifted left or right by one bit)
(xii) Clear (the content of the accumulator or carry flag can be made zero).

ALU does not perform other mathematical operations such as exponential, logarithmic, trigonometric and floating-point operations. These operations are performed by special purpose math processor called floating-point unit (FPU). Modern microprocessors contain an FPU on the microprocessor chip itself (i.e., an on-chip FPU).

***Q.51. Draw logic diagram of ALU that performs AND, OR logic operations and ADD, SUB arithmetic operations.***

***Ans.*** There are number of arithmetic and logic operations that a computer can perform. These are add, subtract, multiply, divide and logic operations or comparisons such as less than, equal to or greater than.

**Logic Diagram of ALU** – Functionally, an ALU can be divided into two segments – the arithmetic unit and the logic unit. The arithmetic unit performs typical arithmetic operations such as addition, subtraction, and increment or decrement by 1. Usually, the operands involved may be signed or unsigned integers. In some cases, however, an arithmetic unit must handle 4-bit binary coded decimal (BCD) numbers and floating-point numbers. Therefore, this unit must include necessary electronics to manipulate these data types. As the name implies, the logic unit contains hardware elements that perform typical operations such as Boolean NOT and OR. Here, the design of a simple ALU using typical combinational elements such as gates, multiplexers, and a 4-bit parallel adder is discussed. For this approach, first an arithmetic unit and a logic unit are designed separately, then they are combined to obtain an ALU.

For the first step, a two-function arithmetic unit as shown in fig. 2.48, is designed. The key element of this system is a 4-bit parallel adder. The multiplexer

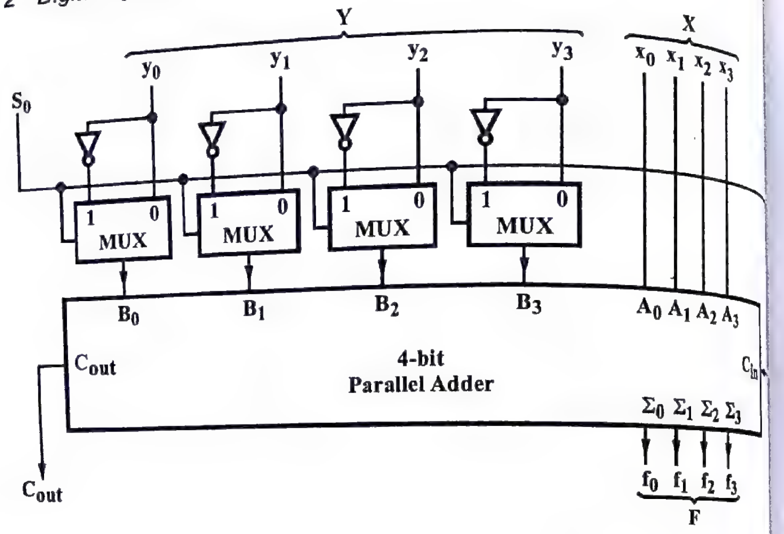


*Fig. 2.48 Organization of an Arithmetic Unit*

selects either Y or Y' for the 3-input of the parallel adder. In particul $S_0 = 0$, then B = Y, otherwise B = Y'. Since the selection input ($S_0$) controls the input carry ($C_{in}$), the following results –

If $S_0 = 0$ then $F = X \text{ plus } Y$

else $F = X \text{ plus } Y' \text{ plus } 1$

$= X \text{ minus } Y$

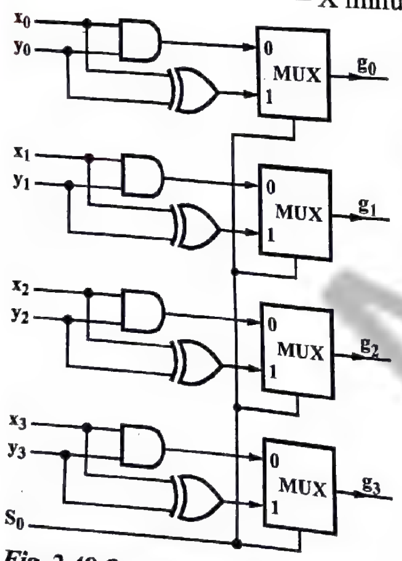


*Fig. 2.49 Organization of a Logic Unit*

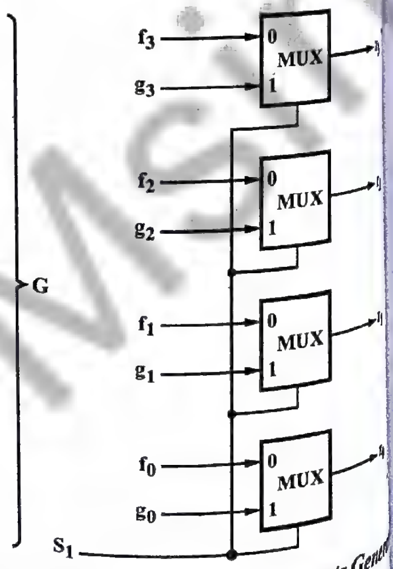


*Fig. 2.50 Combining the Outputs Gen by the Arithmetic and Logic Unit*

For the second step, a two-function logic unit, as shown in fig. 2.49, is designed. From this figure, it can be seen that when $S_0 = 0$, the output G = X AND Y; otherwise the output $G = X \oplus Y$. Note that from these two Boolean operations, other operations such as NOT and OR can be derived by the following Boolean identities –

$$1 \oplus x = x'$$

$$x \text{ OR } y = x \oplus y \oplus xy$$

Therefore, NOT and OR operations can be obtained by using additional hardware and the circuit of fig. 2.49. The outputs generated by the arithmetic and logic units can be combined by using a set of four multiplexers as shown in fig. 2.50.

From this organization it can be seen that when the select line $S_1 = 1$, the multi plexers select outputs generated by the logic unit; otherwise, the outputs of arithmetic unit are selected. The select line, $S_1$, is referred to as the ***mode input*** since it selects the derived mode of operation. A complete block diagram schematic of this ALU is shown in fig. 2.51.

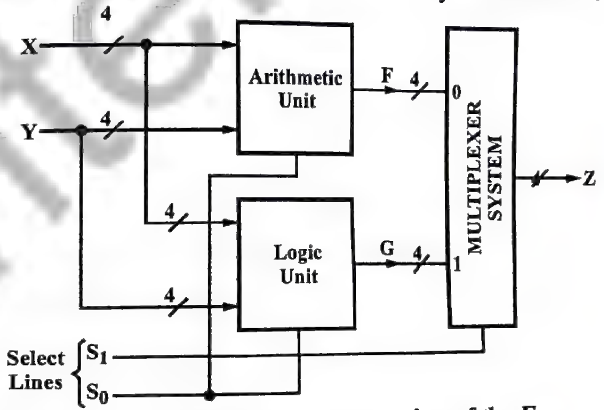


*Fig. 2.51 Schematic Representation of the Four-functions ALU*

| Select Lines | | Output Z | Comment |
|---|---|---|---|
| $S_1$ | $S_2$ | | |
| 0 | 0 | X plus Y | Addition |
| 0 | 1 | X plus Y plus 1 | 2's complement-subtraction |
| 1 | 0 | $X \wedge Y$ | Boolean AND |
| 1 | 1 | $X \oplus Y$ | Exclusive-OR |

*Fig. 2.52 Truth Table Controlling the Operation of the ALU of Fig. 2.51*

The truth table illustrating the operation of this ALU is shown in fig. 2.52.

## NUMERICAL PROBLEMS

***Prob.6. Construct a 4-to-16-line decoder with five 2-to-4-line decoders with enable.*** ***(R.G.P.V., Feb. 2010)***

***Sol.*** To construct a 4×16 decoder from five 2×4 line decoder with enable input terminal is used. The 4×16 decoder with 2×4 decoder is shown in fig. 2.53.

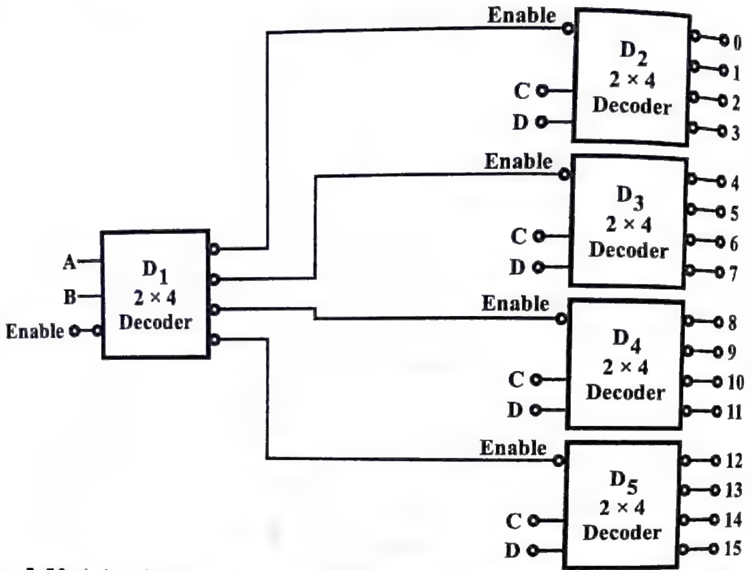


*Fig. 2.53 A 4 × 16 Decoder using Five 2 × 4 Decoder with Enable Input*

***Prob.7. A combinational circuit is defined by the following two functions -***

$$F_1\ (x, y) = \Sigma(0, 3), \quad F_2\ (x, y) = \Sigma(1, 2, 3)$$

***Implement the combinational circuit by means of the decoder and external gates.***

***(R.G.P.V., Nov./Dec. 2007)***

***Sol.*** The truth table for given functions is given in table 2.18.

**Table 2.18 Truth Table for Given Functions**

| *Decimal Number* | *Binary Inputs* | | *Outputs* | |
|---|---|---|---|---|
| | $A_2$ | $A_1$ | $F_1$ | $F_2$ |
| 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 2 | 1 | 0 | 0 | 1 |
| 3 | 1 | 1 | 1 | 1 |

The combinational circuit by means of the decoder and external gates is shown in fig. 2.54.

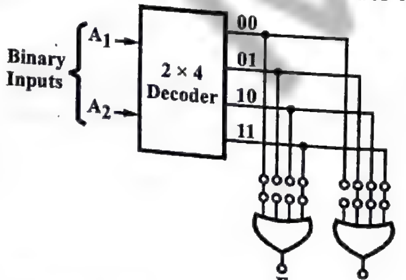


*Fig. 2.54 Combinational Circuit to Obtain Given Functions*

***Prob.8. Design a BCD to excess-3 code converter.***

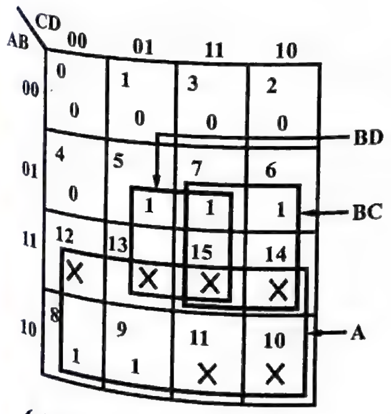
***(R.G.P.V., Dec. 2015, May 2018)***

***Sol.*** The truth table for BCD to Excess-3 code conversion is given in table 2.19.

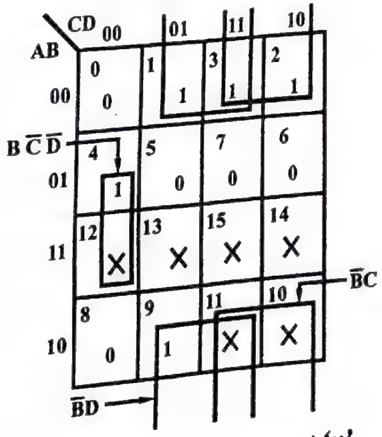
**Table 2.19 Truth Table of BCD to Excess-3 Code Converter**

| *BCD Inputs* | | | | *Excess-3 Code Outputs* | | | |
|---|---|---|---|---|---|---|---|
| *A* | *B* | *C* | *D* | *w* | *x* | *y* | *z* |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |

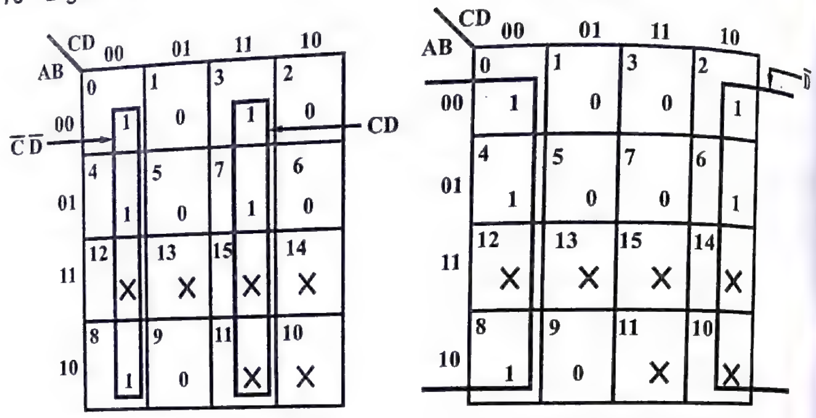
From this truth table, the K-maps in fig. 2.55 are drawn to obtain a simplified Boolean function for each output.



*(a) K-map for Output 'w'*



*(b) K-map for Output 'x'*



(c) K-map for Output 'y' (d) K-map for Output 'z'

*Fig. 2.55 K-maps for BCD to Excess-3 Code Converter*

The minimized expressions obtained from the K-maps in fig. 2.55 are given below –

$$w = A + BC + BD \quad ...(i)$$

$$x = \overline{B}C + \overline{B}D + B\overline{C}\,\overline{D} \quad ...(ii)$$

$$y = CD + \overline{C}\,\overline{D} = C \oplus D \quad ...(iii)$$

$$z = \overline{D} \quad ...(iv)$$

The implementation of BCD to Excess-3 code conversion is illustrated in fig. 2.56.

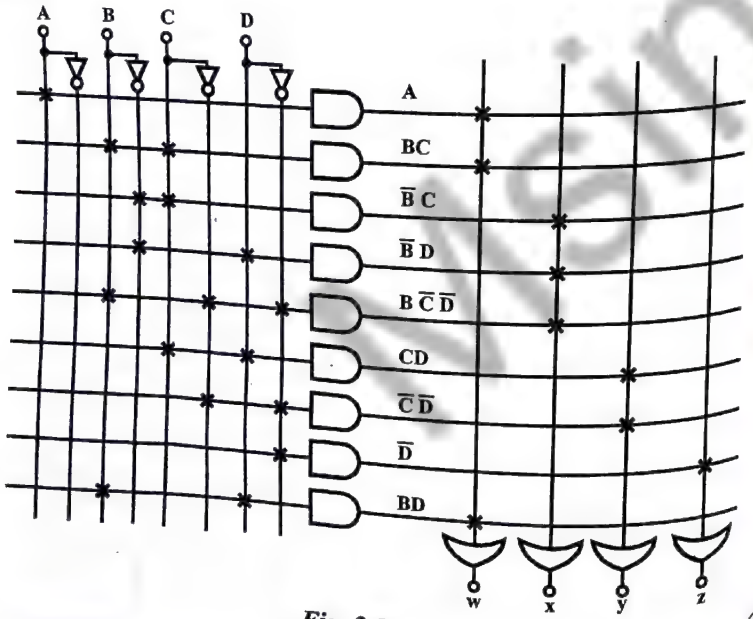


*Fig. 2.56*

# UNIT 3

## SEQUENTIAL LOGIC – FLIP-FLOPS, D, T, S-R, J-K MASTER-SLAVE, RACING CONDITION, EDGE & LEVEL TRIGGERED CIRCUITS

***Q.1. What are sequential circuits ? What is the main difference between the combinational circuits and sequential circuits ? (R.G.P.V., Nov. 2018)***

*Ans.* A block diagram of a sequential circuit is shown in fig. 3.1. It has combinational circuits that take digital signals from external inputs and from outputs of memory elements and produces signals for external outputs and for inputs to memory elements known as excitation.

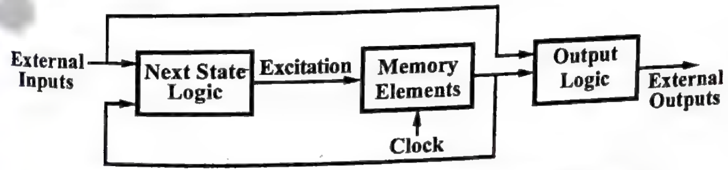


***Fig. 3.1 Block Diagram of a Sequential Circuit***

A memory element is a medium which can be stored or retained one bit of information until necessary and there after these contents can be changed by a new value. In fig. 3.1, the contents of memory elements can be replaced by the outputs of the combinational circuit which are connected to its input.

Sequential circuits are divided into two main categories such as asynchronous sequential circuits, synchronous sequential circuits.

Asynchronous sequential circuits are those circuits whose behaviour depends upon the sequence in which the input signals change. The output will be affected by the change in inputs. Time delay devices are commonly used as memory elements in these circuits. Synchronous sequential circuits are those whose behaviour can be defined from the knowledge of its signal at discrete instants of time. The synchronization is obtained by a timing device which is known as a system clock. This system clock generates a periodic train of pulses as shown in fig. 3.2.

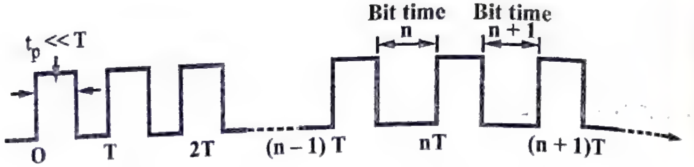


*Fig. 3.2 Periodic Train of Pulses*

Synchronous circuits are very much popular and are also known a clocked sequential circuits. Flip-flops are the memory elements used, whic are capable of storing binary information.

**Difference** – The differences between a combinational circuit and sequential circuit are as follows –

| *S.No.* | *Combinational Circuit* | *Sequential Circuit* |
|---|---|---|
| (i) | It contains no memory elements. | It contains memory elements. |
| (ii) | The present values of its outputs are determined solely by the present values of its inputs. | The present values of its output are determined by the present values of its inputs and it present state. |
| (iii) | Its behaviour is described by the set of output functions. | Its behaviour is described by the set of next-state functions an the set of output functions. |

***Q.2. What is flip-flop ? How many flip-flops are required for storing n-bit of information ?*** ***(R.G.P.V., June 201***

*Ans.* **Flip-flop** – The simplest kind of sequential circuit is a memory ce that has only two states. It can be either 1 or 0. Such two state sequenti circuit is called ***flip-flop*** because they flip from one state to another and the flop back. A flip-flop is also known as ***bistable multivibrator, latch*** or *toggl*

There are two S-R type of gated flip-flops –

(i) Gated S-R Flip-Flop (ii) Gated D Flip-Flop.

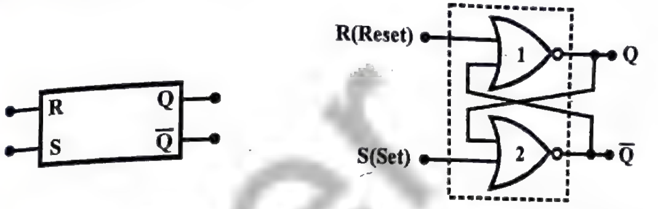
For storing n-bit of information n flip-flops are required because one flip flop store 1-bit of information at a time.

***Q.3. What is set-reset (S-R) flip-flop ? Explain S-R flip-flop with NO gates.***

***Or***

***Explain the working of R-S flip-flop.*** ***(R.G.P.V., Dec. 201***

*Ans.* **Set-Reset (S-R) Flip-flop or (R-S) Filp-flop** – The S-R flip-flo has two inputs, namely Set (S) and Reset (R), and two outputs Q and $\overline{Q}$. Th two outputs are complement to each other. The S-R flip-flop can be easily implemented using NOR gates or NAND gates. The block diagram of S-R flip-flop is shown in fig. 3.3 (a).



***(a) Block Diagram*** ***(b) NOR-based S-R Flip-flop***

***Fig. 3.3 S-R Flip-flop***

**NOR-based S-R Flip-flop** – The S-R flip-flop can be easily constructed using two NOR gates connected back-to-back, as shown in fig. 3.3 (b). The cross-coupled connections from the output of one gate to the input of the other gate constitute a feedback path. For this reason, the circuits are classified as asynchronous sequential circuits. The truth table for the NOR-based S-R flip-flop is shown in table 3.1.

**Table 3.1 NOR-based S-R Flip-flop**

| *Inputs* | | *Outputs* | | *Action* |
|---|---|---|---|---|
| *S* | *R* | $Q_{n+1}$ | $\overline{Q}_{n+1}$ | |
| 0 | 0 | $Q_n$ | $\overline{Q}_n$ | No change |
| 0 | 1 | 0 | 1 | Reset |
| 1 | 0 | 1 | 0 | Set |
| 1 | 1 | ? | ? | Forbidden |

To analyse the circuit of fig. 3.3 (a), one must remember that the output of a NOR gate is 0 if any input is 1 and the output is 1 only when all inputs are 0. From the truth table, it is evident that four possible input combinations exist for the S-R flip-flop. The outputs for these four possible input combinations are described below.

**Case 1** – For S = 0 and R = 0, the flip-flop simply remains in its present state ($Q_n$). That is the next state of the flip-flop ($Q_{n+1}$) is just the present state. In this situation, the next state of the flip-flop will be $Q_{n+1} = 0$ if $Q_n = 0$ and $Q_{n+1} = 1$ if $Q_n = 1$. First, let us assume that $Q_n = 0$ and $\overline{Q}_n = 1$. The inputs of NOR gate-1 are 1 and 0, and therefore its output $Q_{n+1} = 0$. This $Q_{n+1} = 0$ is fed back to NOR gate-2 input thereby producing a 1 at its output; so $\overline{Q}_{n+1} = 1$, as originally assumed.

Next, let us assume that $Q_n = 1$ and $\overline{Q}_n = 0$. This 1 is applied to the i of NOR gate-2 and therefore the output becomes 0 (i.e., $\overline{Q}_{n+1} = 0$). T $Q_{n+1} = 0$ is fed to the input of NOR gate-1, thereby producing a 1 at output; so $\overline{Q}_{n+1} = 1$, as originally assumed. Thus, the condition S = 0 and = 0 will not affect the outputs of flip-flop.

**Case 2** – The second input condition is S = 0 and R = 1. The 1 at t RESET input forces the output of NOR gate-1 Low (i.e., $Q_{n+1} = 0$). No both the inputs of NOR gate-2 are 0 and its output $\overline{Q}_{n+1} = 1$. Thus, the inp condition S = 0 and R = 1 will always ***reset*** the flip-flop to 0. When the res input returns to 0, the flip-flop will remain in the 0 state.

**Case 3** – The third input condition is S = 1 and R = 0, which forces t output of NOR gate-2 LOW i.e., $\overline{Q}_{n+1} = 0$. Now, both the inputs of NO gate-1 are 0, and therefore the output of NOR gate-1 is High i.e., $Q_{n+1} = 1$ Hence, the conditions S = 1 and R = 0 will always ***set*** the flip-flop to 1.

**Case 4** – The last input condition is S = 1 and R = 1. This condition wi produce 0 at the output of both the NOR gates. Hence. $Q_{n+1} = 0$ and $\overline{Q}_{n+1} = 0$ This condition violates the fact that the outputs $Q_{n+1}$ and $\overline{Q}_{n+1}$ are th complements of each other. In normal operation, this condition must be avoide by making sure that 1's are not applied to both inputs simultaneously.

***Q.4. What are flip-flops ? Construct a S-R flip-flop with NAND gat Explain its operation carefully pointing out the illegal conditions.***

***(R.G.P.V., Dec. 201***

***Ans.*** **Flip-flop** – Refer the ans. of Q.2.

**NAND-based $\overline{S}$-$\overline{R}$ Flip-flop** – A basic flip-flop circuit constructed usi cross-coupled NAND gates is shown in fig. 3.4. The operation of NAN $\overline{S}$-$\overline{R}$ flip-flop can be analyzed in the same manner employed for the NOR fli flop. To understand the operation of NAND-based $\overline{S}$-$\overline{R}$ flip-flop, one must remember that a low at any input of a NAND gate will force its output high. The truth table for the NAND-based $\overline{S}$-$\overline{R}$ flip-flop is shown in table 3.2 which is different from that of a NOR-based S-R flip-flop. This flip-flop is called as $\overline{S}$-$\overline{R}$ flip-flop i.e., here $\overline{S}$ = 0 and $\overline{R}$ = 1 will set the flip-flop.

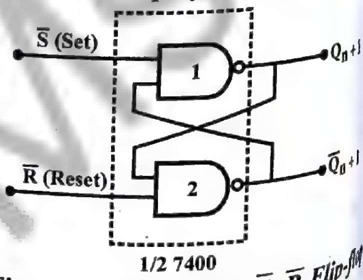


***Fig. 3.4 NAND-based $\overline{S}$-$\overline{R}$ Flip-flo***

**Case 1** – The first condition is $\overline{S} = 0$ and $\overline{R} = 0$. When both inputs g to 0, both outputs go to 1 i.e., $Q_{n+1} = 1$ and $\overline{Q}_{n+1} = 1$. This condition i ambiguous and should not be used.

**Case 2** – The condition $\overline{S} = 0$ and $\overline{R} = 1$ always produces $Q_{n+1} = 1$ regardless of the present state of the flip-flop output. This condition sets the state of the flip-flop i.e., as shown $Q_{n+1} = 1$ and $\overline{Q}_{n+1} = 0$.

**Case 3** – The condition $\overline{S} = 1$ and $\overline{R} = 0$ forces the lower NAND gate output to 1, i.e. $\overline{Q}_{n+1} = 1$. Now both the inputs of upper NAND gate are 1, and therefore the output of upper NAND gate is Low, i.e. $Q_{n+1} = 0$, regardless of the prior state of the flip-flop. This condition ***resets*** (clear) the flip-flop i.e., $Q_{n+1} = 0$ and $\overline{Q}_{n+1} = 1$.

**Case 4** – The last condition $\overline{S} = 1$ and $\overline{R} = 1$, does not affect the state of the flip-flop. It remains in its prior state.

**Table 3.2**

| *Inputs* | | *Outputs* | | *Action* |
|---|---|---|---|---|
| $\overline{S}$ | $\overline{R}$ | $Q_{n+1}$ | $\overline{Q}_{n+1}$ | |
| 0 | 0 | ? | ? | Forbidden |
| 0 | 1 | 1 | 0 | Set |
| 1 | 0 | 0 | 1 | Reset |
| 1 | 1 | $Q_n$ | $\overline{Q}_n$ | No change |

***Q.5. What is a flip-flop ? Explain the principle of operation of R-S flip-flop with truth table.*** ***(R.G.P.V., Nov. 2018)***

***Or***

***What is a flip-flop ? Explain with a suitable example.***

***(R.G.P.V., Dec. 2017)***

***Ans.*** Refer the ans. of Q.2, Q.3 and Q.4.

***Q.6. Explain the function of D flip-flop using a suitable diagram and discuss how it works as a latch.*** ***[R.G.P.V., June 2002 (EL/ET)]***

***Ans.*** The edge-triggered D flip-flop uses an edge detector circuit to ensure that the output will respond to the D input only when the active transition of the clock occurs. If this edge detector is not used and the resultant circuit operates some what differently. It is called a D latch, as shown in fig. 3.5. This circuit contains the NAND latch and the guiding NAND gates $G_1$ and $G_2$ without the edge detector circuit. The common input to the guiding gates is

**Table 3.3 Truth Table of D Latch**

| *Inputs* | | *Output* |
|---|---|---|
| *EN* | *D* | *Q* |
| 0 | × | $Q_0$ (No change) |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

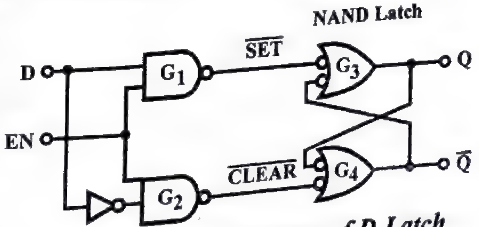


***Fig. 3.5 Logic Diagram of D Latch***

called an enable input rather than a clock input, since its effect on the Q a $\overline{Q}$ outputs is not restricted to occurring only on its transitions.

The logic symbol of D latch is shown in fig. 3.6. When the EN input is hi and the D input will produce a low at either $\overline{SET}$ or the $\overline{CLEAR}$ inputs of t NAND latch to cause Q to become the same level as D. If D changes while E input is high then the output Q will follow the changes exactly. In other wo while EN = 1, the output Q will look exactly like D. In this mode the D latch said to be transparent. When EN input goes low, the D input is inhibited fro affecting the NAND latch since the outputs of both guiding gates will b held high. Thus the output Q and $\overline{Q}$ will stay at whatever level they had only before EN went low. In other words the outputs are latched to their current level and cannot change while EN is low even if D changes.

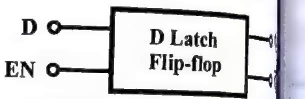


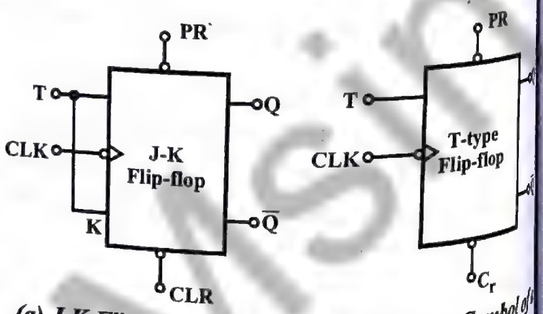
*Fig. 3.6 Logic Symbol of DLatch*

From the logic symbol for the D latch we note that even though the E input operates much like the CLK input of an edge triggered flip-flop and ther is no small triangle on the EN input. Since the small triangle symbol is use strictly for inputs that can cause an output change only when a transitio occurs and the D latch is not edge triggered.

***Q.7. Discuss the working and application of T flip-flop. Also draw i state diagram and write its characteristic equation.***

***Ans.*** **Working –**

If J = K, in a J-K flip flop, the resulting flip-flop is known as a T-type flip-flop. It is shown in fig. 3.7. In this, only one input is present which is known as T input. From the truth table which is shown in table 3.4, it is clear that if T = 1, it works as a toggle switch. The output Q changes, for every clock pulse.



*(a) J-K Flip-Flop Converted into a T Flip-flop*

*(b) Logic Symbol of T Flip-flop*

*Fig. 3.7*

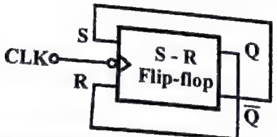


*Fig. 3.8 S-R Flip-flop as a Toggle Switch*

**Table 3.4 Truth Table of T Flip-flop**

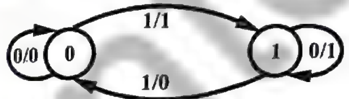
| *Input* $T_n$ | *Output* $Q_{n+1}$ |
|---|---|
| 0 | $Q_n$ |
| 1 | $\overline{Q}_n$ |

An R-S flip flop cannot be converted into a T-type flip flop because R = S = 1 is not permitted. However, the circuit of fig. 3.8 works as a toggle switch, that is, with every clock pulse, the output Q changes.

**Applications of Flip-flop –** Some of the common applications of flip-flops are as –

(i) It can be used as a counter.
(ii) It is used as a building block in sequential circuit such as registers.
(iii) It can be used to bounce-elimination key.
(iv) Flip-flop are used in memory element.
(v) It can be used to latch and delay element.

**State Diagram and Excitation Table –**



**State Diagram**

**T-excitation Table**

| $Q_n$ | $Q_{nt}$ | $T$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

**Characteristic Equation of Flip-flop –**

| *Flip - flop* | *Characteristic Equation* |
|---|---|
| SR | $Q_{n+1} = S + \overline{R}Q_n$ |
| D | $Q_{n+1} = D$ |
| JK | $Q_{n+1} = J\overline{Q}_n + \overline{K}Q_n$ |
| T | $Q_{n+1} = T\overline{Q}_n + \overline{T}Q_n$ |

***Q.8. Draw the logic diagram, construct the excitation table and given the characteristic equation and explain the working of a J-K flip-flop.***

***(R.G.P.V., June 2004)***

***Ans.*** Fig. 3.9 shows the logic diagram of J-K flip-flop using S-R flip-flop. In the state of an S-R flip-flop when S = R = 1 can be eliminated by converting it into a J-K flip-flop. The data inputs are J and K which are ANDed with Q and

**Table 3.5 Excitation Table for J-K Flip-flop**

| *Present State* $Q_n$ | *Inputs* *J* | *K* | *Next State* $Q_{n+1}$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

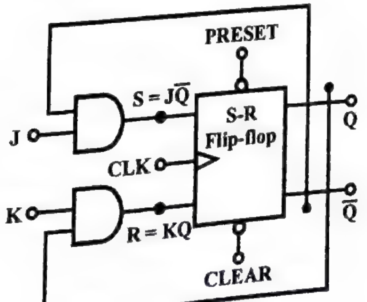


*Fig. 3.9 J-K Flip-flop using S-R Flip-flop*

$\overline{Q}$, respectively, to obtain S and R inputs as –

$$S = J\overline{Q}$$
$$R = KQ$$

The excitation table for J-K flip-flop is given in table 3.5.

Now we obtain the J-K flip-flop using NAND gates as shown fig. 3.10. There are cross connections from output to J-K inputs. The $\overline{Q}$ connected to J input while Q is connected to K input. The truth table of J-K flip-flop is given in table 3.6.

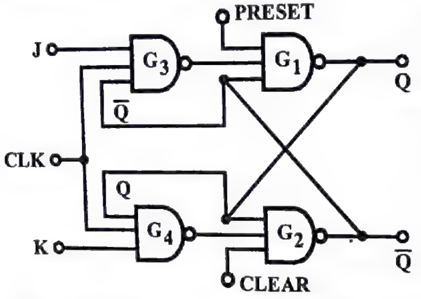


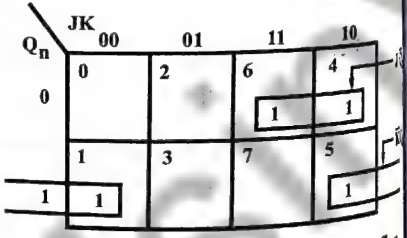
*Fig. 3.10 J-K Flip-flop using Nand Gates*

**Table 3.6 Truth Table for J-K Flip**

| *Inputs* | | *Output* |
|---|---|---|
| *J* | *K* | $Q_{n+1}$ |
| 0 | 0 | $Q_n$ (No change) |
| 0 | 1 | 0 (Reset) |
| 1 | 0 | 1 (set) |
| 1 | 1 | $\overline{Q}_n$ (Toggle) |

From the excitation table in table 3.5, a K-map for the next state transition ($Q_{n+1}$) can be drawn as depicted in fig. 3.11.

From the K-map in fig. 3.11, the minimized characteristic equation of J-K flip-flop can be written as –



$$Q_{n+1} = J\overline{Q}_n + \overline{K}Q_n$$

*Fig. 3.11 K-map for J-K Flip-flop*

**Working** – When J = K = 0, the inputs to the basic flip-flop are S = and R = 0. This condition forces the flip-flop into the same state. If J = K = 1 and the previous state of the flip-flop is set, i.e., $Q_n = 1$, $\overline{Q}_n = 0$, the S = 0 and R = 1. The flip-flop resets on the application of a clock pulse. J = 1, K = 0 and the previous state of the flip-flop is reset, i.e., $Q_n = 0$ an $\overline{Q}_n = 1$, then S = 1 and R = 0. The flip-flop will set on the application of clock pulse. If J = 1, K = 1 and the previous state of the flip-flop is a se i.e., $Q_n = 1$, $\overline{Q}_n = 0$, then S = 0 and R = 1. The flip-flop reset on the application of a clock pulse, i.e., the flip-flop toggles from set state to re state.

***Q.9. What is the race-around condition ? How can this be solved using edge triggering and master-slave flip-flops ?*** ***(R.G.P.V., Dec. 2011)***

*Ans.* **Race-around Condition –** In an S-R flip-flop which is eliminated in a J-K flip-flop by using the feedback connection from the outputs to the inputs of the gates, the difficulty of S = R = 1 being not allowed. For example, consider that the inputs are J = K = 1 and Q = 0. A pulse is shown in fig. 3.12 is applied at the clock input. After a time period Δt equal to the propagation delay through two NAND gates in series, the output will change to Q = 1. Now we have J = K = 1 and Q = 1 and after another time interval of Δt the output will change back to Q = 0. Thus, we note that for the duration $t_p$ of the clock pulse, the output will oscillate back and forth between 0 and 1. At the end of the clock pulse, the value of output Q is uncertain. This situation is referred to as the race-around condition.

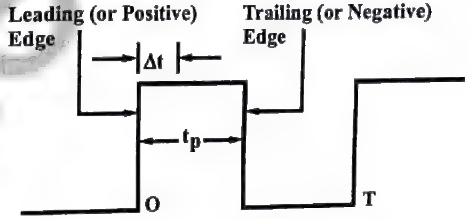


*Fig. 3.12 A Clock Pulse*

The race-around condition can be avoided if $t_p < \Delta t < T$. However, it may be difficult to satisfy this inequality because of very small propagation delays in ICs.

**Edge Triggering –** In case of the pulse-triggered flip-flop one of the major disadvantage is that the J and K leads must be held constant while the clock is active. The edge triggered flip-flop overcomes this problem by activating on one of the edge of the clock pulse.

Fig. 3.13 is a simplified logic diagram of the J-K edge-triggered flip-flop, which triggers on the falling edge of its clock pulse. Assume Q is low, J high and K low and the clock low. When the clock goes high, gate A output will go low and gate B high. Since $\overline{Q}$ is high, gate C will AND, causing Q to remain low.

When the clock goes low, gate C will be disabled, causing Q to go high. This will cause F to AND and $\overline{Q}$ to go low. The action taken by the GH FF depends on what AB was prior to the falling edge of the clockpulse.

The edge triggered device overcomes one problem, it can be fed through a Schmitt trigger to square it up.

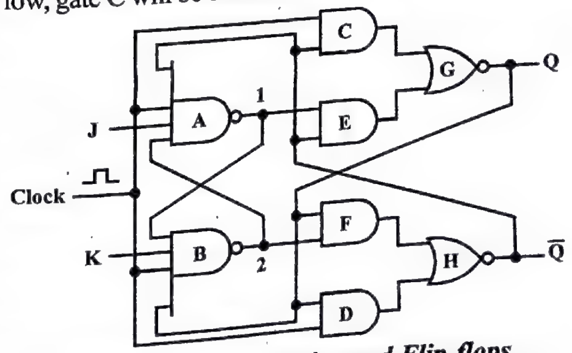


*Fig. 3.13 JK Edge-Triggered Flip-flops*

**Master Slave Flip-flop** – The master-slave J-K flip-flop is a cascade of two S-R flip-flops with feedback from the outputs of the second to the inputs of the first as depicted in fig. 3.14. The positive clock pulses are applied to the first flip-flop and the clock pulses are inverted before these are applied to the second flip-flop.

If CLK = 1, the first flip-flop is enabled and the outputs $Q_M$ and $\overline{Q}_M$ respond to the inputs J and K. At this time, the second flip-flop is inhibited because its clock is low (or $\overline{CLK} = 0$). If CLK goes low (or $\overline{CLK} = 1$), the first flip-flop is inhibited and the second flip-flop is enabled. Thus, the outputs Q and $\overline{Q}$ follow the outputs $Q_M$ and $\overline{Q}_M$, respectively. Since the second flip-flop simply follows the first one, it is referred to as the slave and first one as the master. Thus, this configuration is referred to as master-slave flip-flop.

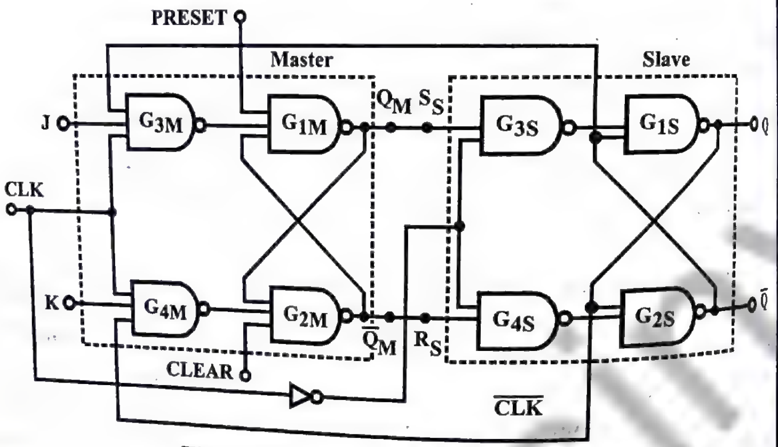


*Fig. 3.14 A Master-slave J-K Flip-flop*

In the master-slave J-K flip-flop circuit, the inputs to the gates $G_{3M}$ and $G_{4M}$ do not change during the clock pulse, thus the race-around condition does not exist. The state of the master-slave flip-flop changes at the negative transition of the clock pulse. The logic symbol of a M-S flip-flop is illustrated in fig. 3.15. At the clock input terminal, the symbol '>' is used to illustrate that the output changes when the clock makes a transition and the accompanying bubble signifies negative transistion.

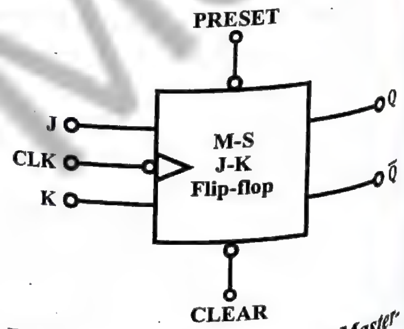


*Fig. 3.15 Logic Symbol of a Master-slave J-K Flip-flop*

***Q.10. What are the various methods used for triggering of flip-flops ?***

***Ans.*** Synchronous bistable devices are known as flip-flops. Synchronous means that the changes in the output obtain at a certained point on a triggering input called the clock. Depend on the particular interval or point in the clock during or at which triggering of flip-flop obtains, it can be divided into two types –

(i) Level triggering (ii) Edge triggering.

We know that the clock input triggers the flip-flop, when the clock pulse goes high and the flip-flop is said to be level triggered flip-flop. Edge triggered means that the flip-flop changes its state of the clock pulse and is sensitive to its inputs only at this transition of the clock.

***Q.11. How a digital circuit is edge triggered ?***

***Ans.*** The clock signal is distributed to all parts of the system and most of the system outputs can change state only when the clock makes a transition. The term edge-triggered means that the flip-flop changes its state either at the positive edge (rising or leading edge) or at the negative edge (falling or trailing edge) of the clock pulse and is sensitive to its inputs only at this transition of the clock. As shown in fig. 3.16, a positive transition is defined as the positive edge and a negative transition as the negative edge. This definition applies also to negative pulses.

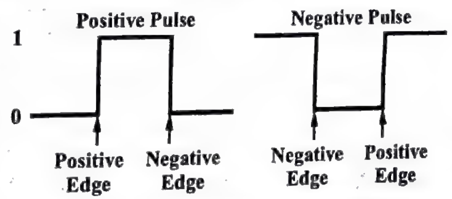
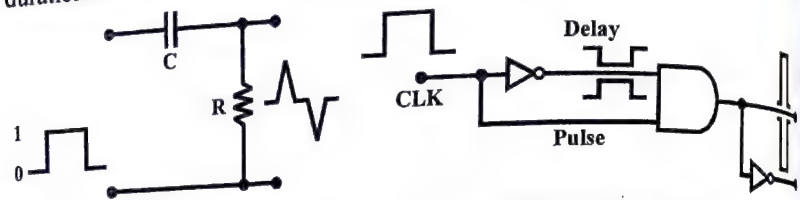


*Fig. 3.16 Definition of Clock Pulse Transition*

Positive-edge triggering is shown by a "triangle" at the clock terminal of the flip-flop. Negative-edge triggering is shown by a 'triangle' with a bubble at the clock terminal of the flip-flop.

One way to make the flip-flop respond only to a pulse transition is to use capacitive coupling. A resistor-capacitor circuit must be inserted in the clock input of the flip-flop as shown in fig. 3.17 (a). The RC time constant is much smaller than the clock's pulse width by deliberate design. Because of this, the capacitor can charge fully when the clock goes high. This exponential charging of capacitor gives a narrow positive voltage spike across the resistor. Later, the trailing edge of the pulse results in a narrow negative spike. A R-C differentiator circuit is the circuit which generates a spike in response to a momentary change of input signal. Edge triggering is obtained by designing the flip-flop to avoid one spike and trigger on the occurrence of the other spike.

A second type of a pulse transition detector is shown in fig. 3.17 (b). There is a small delay on one input of the AND gate due to propagation delay of the NOT gate. So, the inverted clock pulse arrives at the gate input a few nano seconds after the true clock pulse. This gives an output spike with a time duration of only a few nanoseconds.

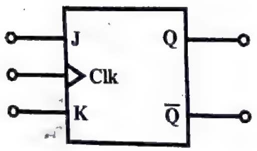


*(a) R-C Differentiator Circuit* *(b) A Type of Pulse Transition Detector*

*Fig. 3.17*

***Q.12. Explain the working of edge triggered J-K flip-flop with the help of its truth table.***

***Ans.*** The functioning of the J-K flip-flop is similar to that of the S-R flip-flop, except that it has no invalid state like that of the S-R flip-flop. Fig. 3.18 shows the logic symbol and the truth table for a positive edge-triggered J-K flip-flop.



*(a) Logic Symbol*

| *Inputs* *J* | *K* | *Clock* | *Outputs* *Q* | *Comments* |
|---|---|---|---|---|
| 0 | 0 | ↑ | $Q_0$ | No change |
| 0 | 1 | ↑ | 0 | Reset |
| 1 | 0 | ↑ | 1 | Set |
| 1 | 1 | ↑ | $Q_0$ | Toggle |

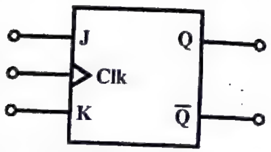
*(b) Truth Table*

*Fig. 3.18 Positive Edge-triggered J-K Flip-flop*

When J = 0 and K = 0, no change of state takes place even if a clock pulse is applied.

When J = 0 and K = 1, the flip-flop resets at the positive-going edge of the clock pulse.

When J = 1 and K = 0, the flip-flop sets at the positive-going edge of the clock pulse.



*(a) Logic Symbol*

| *Inputs* *J* | *K* | *Clock* | *Outputs* *Q* | *Comments* |
|---|---|---|---|---|
| 0 | 0 | ↓ | $Q_0$ | No change |
| 0 | 1 | ↓ | 0 | Reset |
| 1 | 0 | ↓ | 1 | Set |
| 1 | 1 | ↓ | $Q_0$ | Toggle |

*(b)*

*Fig. 3.19 Negative Edge-triggered J-K Flip-flop*

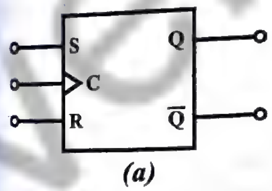
When J = 1 and K = 1, the flip-flop toggles, i.e. goes to the opposite state at the positive-going edge of the clock pulse. In this mode, the flip-flop toggles or changes state for each occurrence of the positive-going edge of the clock pulse.

The logic symbol and the truth table of a negative edge-triggered J-K flip-flop are shown in fig. 3.19.

***Q.13. Explain working of edge triggered RS flip-flop. Write also its excitation table.***

***Ans.*** The S and R inputs of the S-R flip-flop are called the synchronous control inputs because data on these inputs affect the flip-flops output only on the triggering edge of the clock pulse. Without a clock pulse, the S and R inputs cannot affect the output.

Fig. 3.20 (a) and (b) show the logic symbol and the truth table for a positive edge-triggered S-R flip-flop.



*(a)*

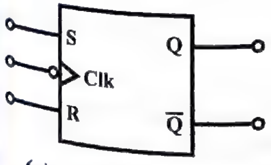
| *Inputs* *S* | *R* | *Clock* | *Outputs* *Q* | *Comments* |
|---|---|---|---|---|
| 0 | 0 | ↑ | $Q_0$ | No change |
| 0 | 1 | ↑ | 0 | Reset |
| 1 | 0 | ↑ | 1 | Set |
| 1 | 1 | ↑ | ? | Invalid |

*(b)*

*Fig. 3.20 Positive Edge-triggered S-R Flip-flop*

When S is LOW and R is HIGH, the Q output goes LOW on the positive-going edge of the clock pulse and the flip-flop is RESET, i.e. cleared.

When S is HIGH and R is LOW, the Q output goes HIGH on the positive-going edge of the clock pulse and the flip-flop is set. When both S and R are LOW, the output does not change from its prior state. When both S and R are HIGH simultaneously, an invalid condition exists. The basic operation described above is illustrated in fig. 3.21 (a). Fig. 3.21 (b) shows the logic symbol and the truth table of a negative edge-triggered S-R flip-flop. This flip-flop will trigger only when the clock input goes from 1 to 0.



*(a) Logic Symbol*

| *Inputs* *S* | *R* | *Clock* | *Outputs* *Q* | *Comments* |
|---|---|---|---|---|
| 0 | 0 | ↓ | $Q_0$ | No change |
| 0 | 1 | ↓ | 0 | Reset |
| 1 | 0 | ↓ | 1 | Set |
| 1 | 1 | ↓ | ? | Invalid |

*(b) Truth Table*

*Fig. 3.21 Negative Edge-triggered S-R Flip-flop*

**Q.14. What do you mean by a level triggered flip-flop ? How does it differ from an edge triggered flip-flop ?**

***Ans.*** Clocked S-R flip flop was shown that the clock input triggers the flip-flop, i.e. enables the flip-flop, when the clock pulse goes HIGH, and flip-flop is said to be level triggered flip-flop. Since the flip-flop changes state when the clock is positive, it is termed as positive level triggered flip-flop. If a NOT gate is introduced in between the clock input and the input of AND gate (a), the flip-flop changes its state only when the clock is negative (i.e. when clock = LOW), and it is called negative level triggered flip-flop.

The main drawback of level triggering is that, as long as the clock is positive or negative, the flip-flop changes its state more than once or many times for change in inputs. If the inputs are made stable for the entire clock duration, the output changes only once. On the other band, if the frequency of input change is higher than the input clock frequency, the output of the flip-flop undergoes multiple changes when the clock is positive or negative. When clock becomes unasserted (i.e. clock = 0), the output of the flip-flop reflects last change in its inputs. This can be overcome in master-slave flip-flops or edge-triggered flip-flops where the flip-flop change state only once for a clock.

Also refer the ans. of Q.11.

**Q.15. Explain the applications of flip-flops.**

***Ans.*** There are a large number of applications of flip-flops. Some of the basic applications are parallel data storage, serial data storage, transfer of data, frequency division, counting, parallel to serial conversion, synchronizing the effect of asynchronous data, detection of an input sequence etc.

***(i) Parallel Data Storage*** – A group of flip-flops is called a register. To store a data of N bits, N flip-flops are required. Since the data is available in parallel form i.e., all bits are present at a time, these bits may be made available at the input terminals of the flip-flops and when a clock pulse is applied to all the flip-flops simultaneously, these bits will be transferred to the outputs of the flip-flops and the flip-flops (register) then store the data.

***(ii) Serial Data Storage*** – To store a data of N bits available in serial form, N number of flip-flops are connected in cascade. The clock signal is connected to all the flip-flops. The serial data is applied to the input terminal of the first flip-flop. Each clock pulse transfers the input to output. So after N clock pulses the register (group of flip-flops) contains the data and then stores it.

***(iii) Transfer of Data*** – Data stored in flip-flops may be transferred out in a serial fashion i.e., bit-by-bit from the output of one flip-flop or may be transferred out in parallel form i.e., all bits at a time from the outputs of each of the flip-flops.

***(iv) Serial to Parallel Conversion*** – To convert the data available in serial form into parallel form the serial data are first entered and stored in a serial in parallel-out shift register and then, since the data are available simultaneously at the outputs of the flip-flops, the data may be taken out parallely. To convert an N-bit serial data into parallel form, N-flip-flops are required. N clock pulses are required to enter the data in serial form and one clock pulse is required to shift the data out in parallel form.

***(v) Counting*** – A number of flip-flops may be connected in a particular fashion to count the pulses electronically. One flip-flop can count up to 2 pulses, two flip-flops can count up to $2^2 = 4$ pulses. In general, N flip-flops can count up to $2^N$ pulses. In a simple counter, all the flip-flops are connected in toggle mode. The clock pulses are applied to the first flip-flop and the clock terminal of each subsequent flip-flop is connected to the output of the previous flip-flop. Feedback may be used to count up or down or up/down.

***(vi) Frequency Division*** – Flip-flops may be used to divide the input signal frequency by any number. A single flip-flop may be used to divide the input frequency by 2. Two flip-flops may be used to divide frequency by 4. In general, N flip-flops may be used to divide the input frequency by $2^N$. If N flip-flops are connected as ripple counter and if the input signal of frequency f is fed to the first flip-flop, the output of this flip-flop will be of frequency f/2, the output of the second flip-flop will be of frequency f/4 and so on.

## SHIFT REGISTERS, ASYNCHRONOUS AND SYNCHRONOUS COUNTERS, THEIR TYPES AND STATE DIAGRAMS

**Q.16. What is a shift register ? Explain.**

***(R.G.P.V., June 2008, 2009, 2010, Dec. 2017)***

***Or***

**Discuss the shift registers.** ***(R.G.P.V., June 2011)***

***Ans.*** A register is simply a group of flip-flops that can be used to store a binary number. There must be one flip-flop for each bit in the binary number. For instance, a register used to store an 8-bit binary number must have eight flip-flops. A group of flip-flops connected to provide shifting of its binary information either to the right or both of these functions is called a shift register. The bits in a binary number can be moved from one place to another in either of two ways. The first method involves shifting the data 1-bit at a time in a serial manner starting with either the most significant bit (MSB) or the least significant bit (LSB). This method referred to as serial shifting. The second method involves shifting, such as serial-in-serial-out, serial-in-parallel-out, parallel-in-serial-out.

***Q.17. What is shift register ? Mention some applications of shift registers.*** ***(R.G.P.V., June 201***

***Ans.*** **Shift Register** – Refer the ans. of Q.16.

**Applications** – Shift registers are used in digital systems for temporary storage of information, data manipulation and transferring. In addition, they are used in counting circuits, such as simple counters, variable modulo counters, up/down counters and increment counters.

***Q.18. Define shift registers and explain serial transfer from register A to register B.*** ***(R.G.P.V., June 201***

***Ans.*** **Shift Register** – Refer the ans. of Q.16.

Fig. 3.22 shows the serial transfer of information from register A to register B.

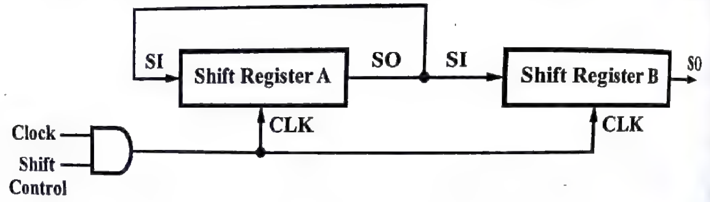


***Fig. 3.22***

In fig. 3.22, register B serial input is connected to serial output of register A. The information in register A is made to circulate by connecting the serial O/P to its serial input to stop the loss of information stored in the source register. Its serial output is shifted the primary content of register B and is lost unless it is transferred to a third shift register. When and how many times the registers are shifted is determined by shift control input. This is done with an AND gate that permits clock pulses to pass into the CLK terminals only if the shift control is active.

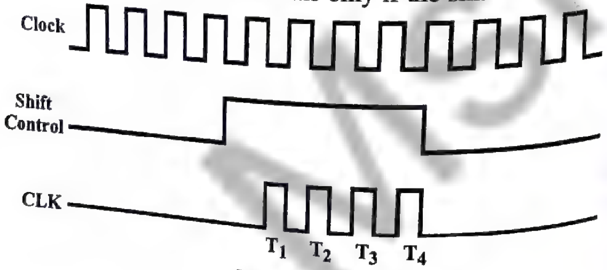


***Fig. 3.23***

For a fixed time of four clock pulses, the control unit which supervises the transfer should be made in such a way that it enables the shift registers by shift control signal as shown in fig. 3.23. The shift control signal is synchronized with the clock and changes value just after the negative edge of the clock. In active state, next four clock pulses find the shift control signal so that the output of the AND gate connected to the clock inputs produces four pulses $T_1$, $T_2$, $T_3$ and $T_4$. Each rising edge of the pulse causes a shift in both registers.

The fourth pulse changes the shift control to 0 and the shift registers are disabled.

**Table 3.7**

| *Timing Pulses* | *Shift Register A* | *Shift Register B* |
|---|---|---|
| Initial value | 1011 | 0010 |
| After $T_1$ | 1101 | 1001 |
| After $T_2$ | 1110 | 1100 |
| After $T_3$ | 0111 | 0110 |
| After $T_4$ | 1011 | 1011 |

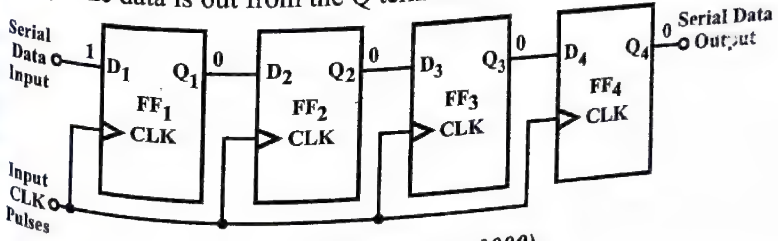
Consider, before shift binary content of A and B is 1011 and 0010 respectively. Table 3.7 shows the serial transfer from A to B gets in four steps. The right most bit of A is shifted into the leftmost bit of B with first pulse $T_1$ and is also circulated into the leftmost position of A. All bits of A and B are shifted one position to the right at the same time. The previous serial output from B in the right most position is lost and its value changes from 0 to 1. The next 3 pulses perform identical operations, shifting the bits of A into B, one at a time. The shift control goes to 0 and both registers A and B have the value 1011 after fourth shift. Hence the content of A is transferred into B, while the content of A remains unchanged.

***Q.19. Explain serial-in-serial-out shift register.***

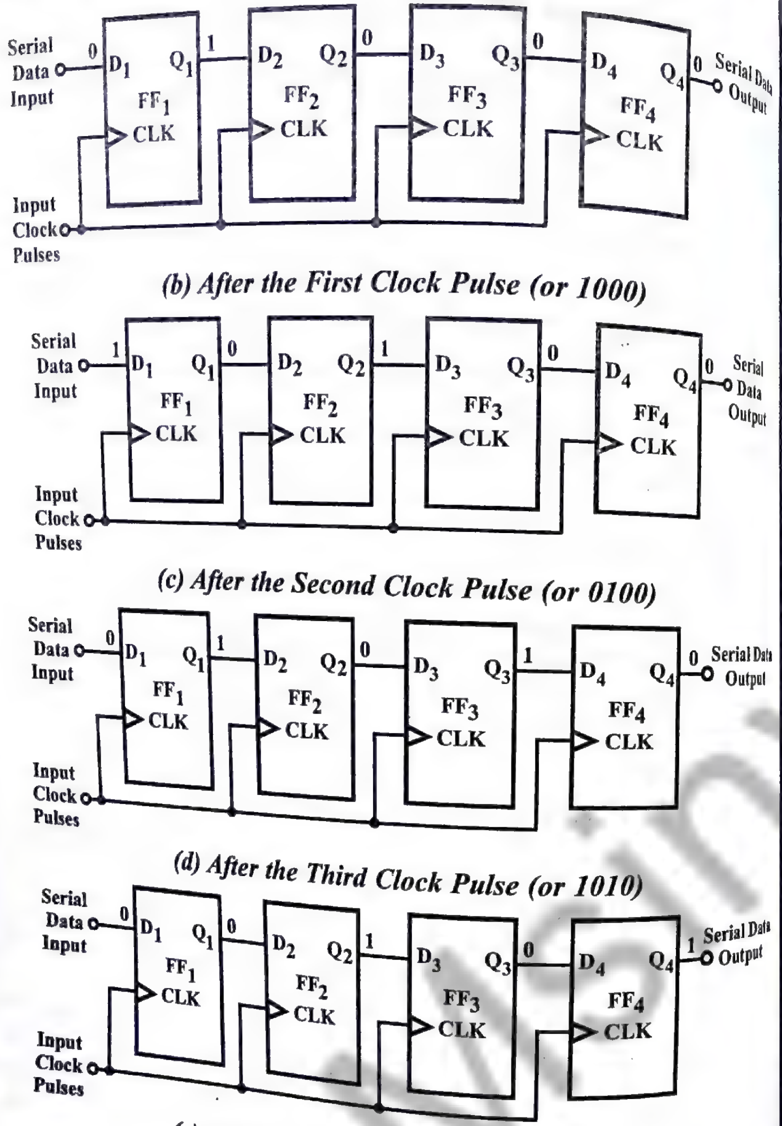
***Ans.*** Serial-in-serial-out shift register accepts data serially, i.e., one bit at a time and also outputs data serially.

The logic diagram of a 4-bit serial-in-serial-out (SISO) shift-right shift register is shown in fig. 3.24. With four stages, i.e., four flip-flops, the register can store up to 4-bits of data.

Serial data is applied at the D input of the first flip-flop. The Q output of the first flip-flop is connected to the D input of the second flip-flop, the Q output of the second flip-flop is connected to the D input of the third flip-flop and Q output of the third flip-flop is connected to the D input of the fourth flip-flop. The data is out from the Q terminal of the last flip-flop.



***(a) Initial States (or 0000)***



*(b) After the First Clock Pulse (or 1000)*

*(c) After the Second Clock Pulse (or 0100)*

*(d) After the Third Clock Pulse (or 1010)*

*(e) After the Fourth Clock Pulse (or 0101)*

***Fig. 3.24 Loading of the Four-bit Serial-in-serial-out Shift Register***

When serial data is transferred into a register, each new bit is clocked into the first flip-flop at the positive-going edge of each clock pulse. The bit that was previously stored by the first flip-flop is transferred to the second flip-flop. The bit that was stored by the second flip-flop is transferred to the third flip-flop and so on. The bit that was stored by the last flip-flop is shifted out. The shifting in the data 0101 serially in the register. Initially reset all the flip-flops, i.e., $Q_1 = 0$, $Q_2 = 0$, $Q_3 = 0$ and $Q_4 = 0$, as given in table 3.8.

**Table 3.8 Shifting in the Data 0101 Serially**

| *After Clock Pulse* | *Serial Input* | *$Q_1$* | *$Q_2$* | *$Q_3$* | *$Q_4$* |
|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 (Initial states) |
| 1 | 0 | 1 | 0 | 0 | 0 |
| 2 | 1 | 0 | 1 | 0 | 0 |
| 3 | 0 | 1 | 0 | 1 | 0 |
| 4 | – | 0 | 1 | 0 | 1 |

The rightmost bit 1 is applied at the $D_1$ input of $FF_1$. At the positive-going edge of the first clock pulse, this 1 is shifted into $FF_1$ and all other flip-flops store their respective bits at the D inputs. Therefore –

$Q_1 = 1$, $Q_2 = 0$, $Q_3 = 0$ and $Q_4 = 0$ (after the first clock pulse)

When 0 is applied, then we get –

$Q_1 = 0$, $Q_2 = 1$, $Q_3 = 0$, $Q_4 = 0$ (after second clock pulse)

When 1 is applied, than we get –

$Q_1 = 1$, $Q_2 = 0$, $Q_3 = 1$, $Q_4 = 0$ (after third clock pulse)

Finally 0 is applied, then we get –

$Q_1 = 0$, $Q_2 = 1$, $Q_3 = 0$, $Q_4 = 1$ (after fourth clock pulse)

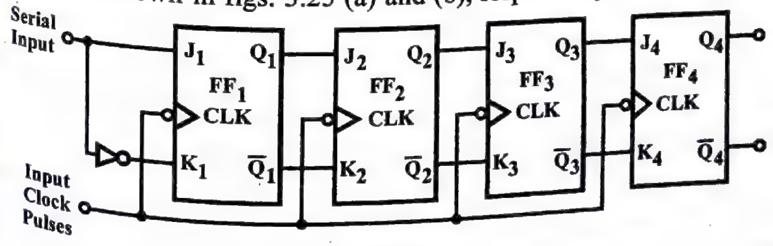
Now completes the serial entry of 0101 into the 4-bit register.

The shifting out of the stored data 0101 serially from the register is shown in table 3.9. It needs four clock pulses to shift out the 4-bit stored data.

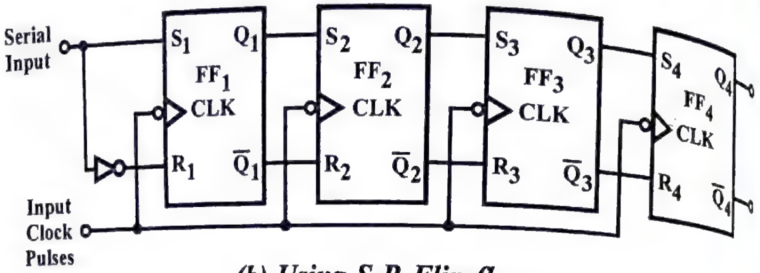
**Table 3.9 Shifting Out the Data 0101 Serially**

| *After Clock Pulse* | *Serial input* | *$Q_1$* | *$Q_2$* | *$Q_3$* | *$Q_4$* |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 1 (Initial states) |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 2 | 0 | 0 | 0 | 0 | 1 |
| 3 | 0 | 0 | 0 | 0 | 0 |
| 4 | – | 0 | 0 | 0 | 0 (Final states) |

This shift register can also be constructed using J-K flip-flops and S-R flip-flops, as shown in figs. 3.25 (a) and (b), respectively.



*(a) Using J-K Flip-flops*



*(b) Using S-R Flip-flops*

***Fig. 3.25 Four-bit Serial-in-serial-out Shift Register***

***Q.20. Discuss briefly about series shift register.*** ***(R.G.P.V., Dec. 2014)***

***Ans.*** Refer the ans. of Q.19.

***Q.21. What is shift register ? Explain serial-in-parallel-out shift register.*** ***(R.G.P.V., Feb. 2010)***

***Ans.*** **Shift Register** – Refer the ans. of Q.16.

**Serial-in-parallel-out Shift Register** – A 4-bit serial-in-parallel-out shift register is shown in fig. 3.26. It consists of one serial input, and outputs are taken from all the flip-flops parallel. In this register, data is shifted in serially but shifted out in parallel. In order to shift the data out in parallel, it is necessary to have all the data available at the outputs at the same time. Once the data is stored, each bit appears on its respective output line and all the bits are available simultaneously, rather than on a bit-by-bit basis as with the serial output.

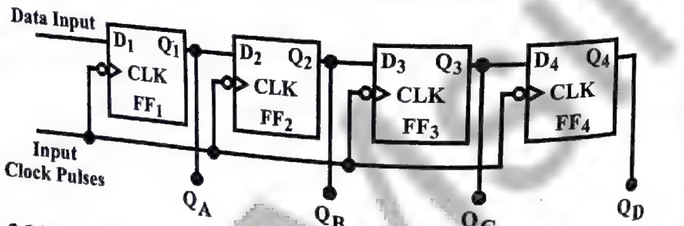


***Fig. 3.26 Logic Diagram of a 4-bit Serial-in-parallel-out Shift Register***

***Q.22. Explain parallel-in-serial-out (PISO).***

***Ans.*** For parallel-in-serial-out shift register, the data bits are entered simultaneously into their respective stages on parallel lines, rather than on a bit-by basis on one line as with serial data inputs, but the data bits are transferred out of the register serially i.e., on a bit-by-bit basis over a single line.

Fig. 3.27 illustrates a 4-bit parallel-in-serial-out shift register using D flip-flops. There are four data lines A, B, C and D through which the data is entered into the register in parallel form. The signal shift/$\overline{\text{Load}}$ allows (a) the data to be entered in parallel form into the register and (b) the data to be shifted out serially from terminal $Q_4$.

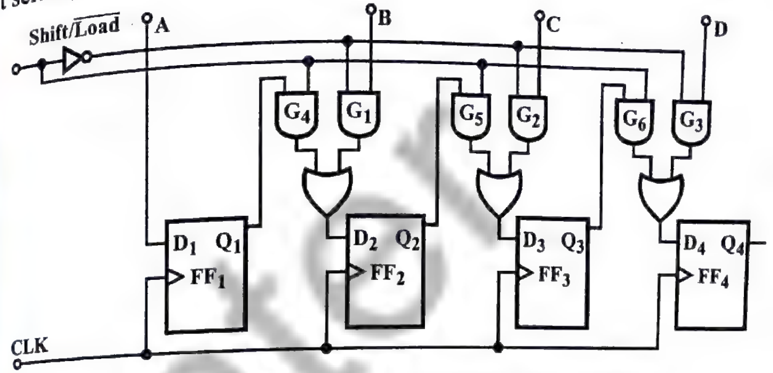


***Fig. 3.27 Logic Diagram of PISO Shift Register***

When shift/$\overline{\text{Load}}$ line is HIGH, gates $G_1$, $G_2$ and $G_3$ are disabled, but gates $G_4$, $G_5$ and $G_6$ are enabled allowing the data bits to shift-right from one stage to the next. When shift/$\overline{\text{Load}}$ line is LOW, gates $G_4$, $G_5$ and $G_6$ are disabled, whereas gates $G_1$, $G_2$ and $G_3$ are enabled allowing the data input to appear at the D inputs of the respective flip-flops. When a clock pulse is applied, these data bits are shifted to the Q output terminals of the flip-flops and therefore, data is inputed in one step. The OR gate allows either the normal shifting operation or the parallel data entry depending on which AND gates are enabled by the level on the shift/$\overline{\text{Load}}$ input.

***Q.23. Explain parallel-in-parallel-out shift register. (R.G.P.V., Dec. 2005)***

***Ans.*** In parallel-in-parallel-out shift register the data is entered into the register in parallel form and also the data is taken out of the register in parallel form. Fig. 3.28 shows a 4-bit parallel-in-parallel-out shift register using D flip-flops.

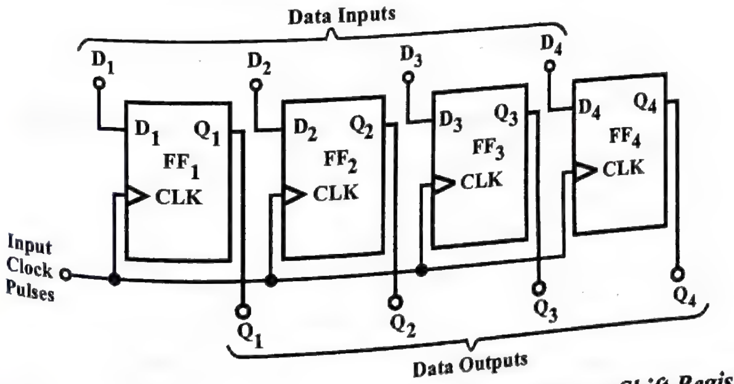


***Fig. 3.28 Logic Diagram of a 4-bit Parallel-in-parallel-out Shift Register***

Data is applied to the D input terminals of the flip-flops. When a clock pulse is applied at the positive-going edge of that pulse, the D inputs are shifted into the Q outputs of the flip-flops. The register now stores the data. The stored data is available instantaneously for shifting out in parallel form. The parallel-in parallel-out shift registers can be used as Read/Write memory registers.

***Q.24. Explain the designing features and working of a 4-bit bidirectional shift register with parallel load.***

***Ans.*** A bidirectional shift register is one in which the data bits can be shifted from left to right or from right to left. Shift registers can be used for converting serial data to parallel data and vice versa. If a parallel load capability is added to a shift register, then data entered in parallel can be taken out in serial fashion by shifting the data stored in the register.

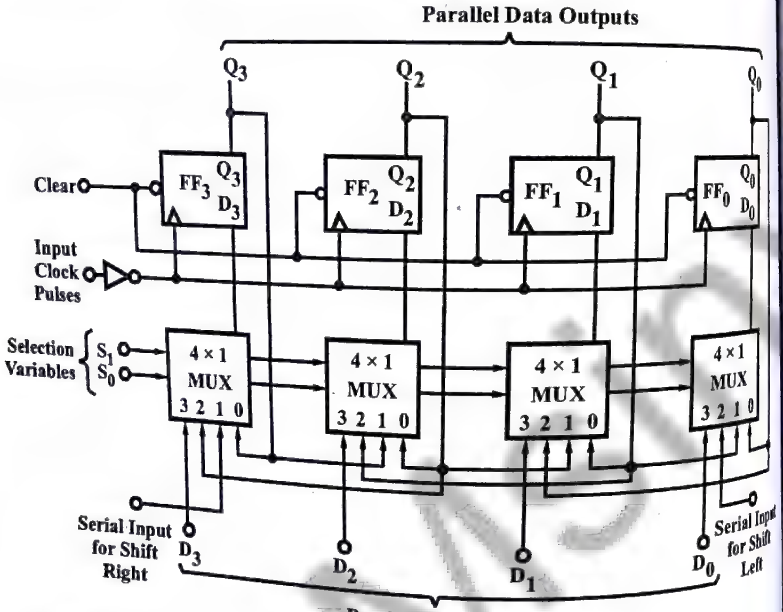


***Fig. 3.29 Four-bit Bidirectional Shift Register with Parallel Load***

Designing features of 4-bit bidirectional shift register with parallel load are as follows –

(i) A clear control to clear the register to zero.

(ii) A input clock pulses to synchronize all operations.

(iii) A shift-right control to enable the shift-right operation and the serial input and output lines associated with the shift right.

(iv) A shift-left control to enable the shift-left operation and the serial input and output lines associated with the shift-left.

(v) A parallel-load control to enable a parallel transfer and the n input lines associated with the parallel transfer.

(vi) There is 'n' parallel output lines.

(vii) A control state that leaves the information in the register unchanged even though clock pulses are continuously applied.

The design of a shift register that has all the functionality from case (i) to case (vii), is as shown in fig. 3.29.

If the register has both shift and parallel load capability, it is known as shift register with parallel load. It consists of four D flip-flops and the four multiplexers (MUX) that are part of the register. All multiplexers have two common selection variables, $S_1$ and $S_0$. When $S_1S_0 = 00$ each MUX selects input 0 and when $S_1S_0 = 01$ input 1 is selected. Similar inputs are selected for the other two inputs to the multiplexers. The $S_1$ and $S_0$ inputs control the mode of operation of the register as specified in the function entries of table 3.10.

**Table 3.10 Function Table for the Register**

| *Mode* | *Control* | *Register* |
|---|---|---|
| $S_1$ | $S_0$ | *Operation* |
| 0 | 0 | No change |
| 0 | 1 | Shift right |
| 1 | 0 | Shift left |
| 1 | 1 | Parallel load |

When $S_1S_0 = 00$, the present value of the register is applied to the D inputs of the flip-flops. This condition forms a path from the output of each flip-flop into the input of the same flip-flop. The next clock pulse transfers into each flip-flop the binary value it held previously and no change of state occurs. When $S_1S_0 = 01$, terminals 1 of the multiplexer inputs have a path to the D inputs of the flip-flops. This causes a shift-right operation, with the serial input transferred into $FF_3$. When $S_1S_0 = 10$, a shift-left operation results, with the other serial input going into $FF_0$. When $S_1S_0 = 11$, the binary information on parallel input lines is transferred into the register simultaneously during the next clock pulse.

***Q.25. What is shift register ? Draw and explain shift left-right register.*** ***(R.G.P.V., May 2018)***

***Ans.*** Refer the ans. of Q.16 and Q.24.

***Q.26. Explain counters.*** ***(R.G.P.V., Dec. 2013)***

***Ans.*** One of the most important operation in digital system is the counting operation. A logic circuit employed to count the number of pulses inputted to it, is called a counter. The pulses may represent some events. Apart from the counting operation, the counter is also responsible for remembering the present number, so that it can go to the next appropriate number in sequence when the next pulse comes. So, storage elements, that is, flip-flops are employed to build counters.

The composition of counters is performed by using flip-flops. A 3-bit counter consisting of three flip-flops is depicted in fig. 3.30. A circuit with n-flip-flops has $2^n$ possible states. Thus, the 3-bit counter can count from decimal 0 to 7.

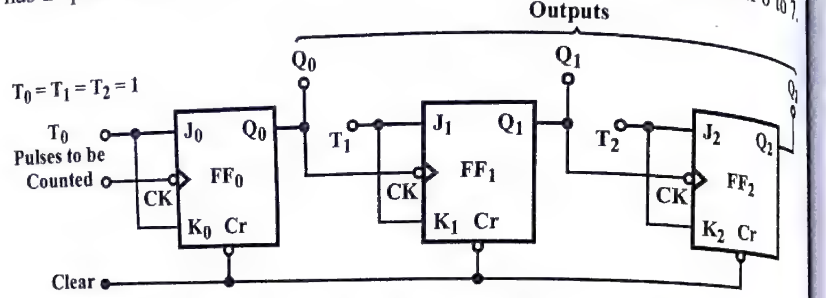


*Fig. 3.30 A 3-bit Counting Flip-flops*

***Q.27. Differentiate between synchronous and asynchronous counters.***
***(R.G.P.V., June 2005, 2012)***

***Ans.*** The differences between synchronous and asynchronous counter are as follows –

| *S.No.* | *Synchronous Counter* | *Asynchronous Counter* |
|---|---|---|
| (i) | In synchronous counter the input clock pulse is applied to all the flip-flops simultaneously. | In asynchronous counter the clocked input is applied to first flip-flop only and other flip-flop are triggered by the output of successive flip-flop. |
| (ii) | The speed of operation is high. | The speed of operation is low. |
| (iii) | All the flip-flops change the state simultaneously. | The change of state of all flip-flops simultaneously is not possible. |

***Q.28. What is counter ? Differentiate synchronous and asynchronous counters.***
***(R.G.P.V., Dec. 2017)***

***Ans.*** Refer the ans. of Q.26 and Q.27.

***Q.29. Write short note on asynchronous counter. (R.G.P.V., June 2009)***

***Ans.*** An asynchronous counter is one in which the flip-flops within the counter do not change states at exactly the same time because they do not have a common clock pulse. All the flip-flops are not under the control of a single clock pulse in this counter. The clock pulse is applied to the first flip-flop, i.e., the least significant bit stage of the counter and the successive flip-flop is triggered by the output of the previous flip-flop. Thus, its speed of operation is limited. As the triggers move through the flip-flops like a ripple, it is called ripple counter. The ripple counter is the simplest type of counter, the easiest to design and needs the least amount of hardware.

Two-bit asynchronous counter is shown in fig. 3.31. Here we note that the clock pulse is applied to the $FF_0$, which is always the least significant bit. The $FF_1$ is triggered by the $\overline{Q}_0$ output of $FF_0$. The $FF_0$ changes state at the positive-going edge of each clock pulse, but $FF_1$ changes state only when triggered by a positive going transition of $\overline{Q}_0$ output of $FF_0$. Because of the inherent propagation delay time through a flip-flop, a transition of the $\overline{Q}_0$ output of $FF_0$ can never occur at exactly the same time. Therefore, the two flip-flops are never triggered simultaneously, so the counter operation is asynchronous.

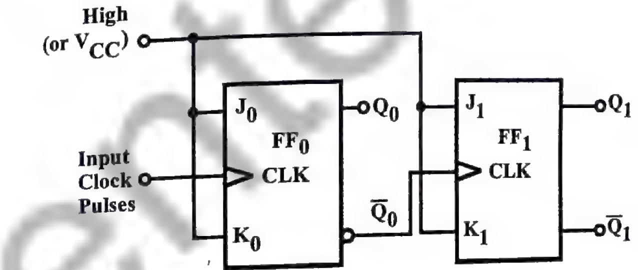


*Fig. 3.31 Two-bit Asynchronous Binary Counter*

Fig. 3.32 illustrates the waveforms of 2-bit asynchronous counter. Both flip-flops are connected for toggle operation (i.e., J = K = 1) and are assumed to be initially reset. The positive-going edge of clock pulse-1 causes the $Q_0$ output of $FF_0$ to go high (or logic-1). At the same time the $\overline{Q}_0$ output goes low (or logic-0), but it has no effect on $FF_1$ because a positive going transition must occur to trigger the flip-flop. After the leading edge of clock pulse-1, output $Q_0 = 1$ and output $Q_1 = 0$. The positive-going edge of clock pulse-2 causes $Q_0$ output to go low. The output $\overline{Q}_0$ goes high and triggers $FF_1$, causing output $Q_1$ to go high. After the leading edge of the clock pulse-2. Outputs $Q_0 = 0$ and $Q_1 = 1$. The positive-going edge of clock pulses-3 causes output $Q_0$ to go high again. The output $\overline{Q}_0$ goes low and has no effect on $FF_1$. Therefore, after the leading edge of clock pulse-3, output $Q_0 = 1$ and

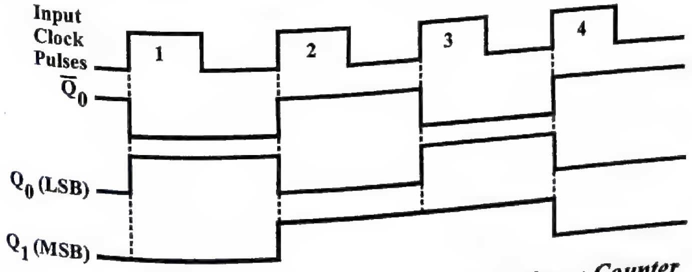


*Fig. 3.32 Waveforms of 2-bit Asynchronous Binary Counter*

$Q_1 = 1$. The positive-going edge of clock pulse-4 causes $Q_0$ output to go low while $\overline{Q}_0$ output goes high and triggers $FF_1$, causing $Q_1$ output to go low. After the leading edge of clock pulse-4, outputs $Q_0 = 0$ and $Q_1 = 0$. The counter now recycled to its original state.

***Q.30. Draw the block diagram of binary ripple counter and explain with suitable waveforms.*** ***(R.G.P.V., June 2007)***

*Ans.* A 4-bit binary ripple counter using clocked J-K flip-flops is shown in fig. 3.33. In a ripple counter, the flip-flop output transition serves as a source for triggering other flip-flops. In other words, the clock pulse input of all flip-flop except the first flip-flop are not triggered by the incoming pulses but rather by the transition occurs in other flip-flops.

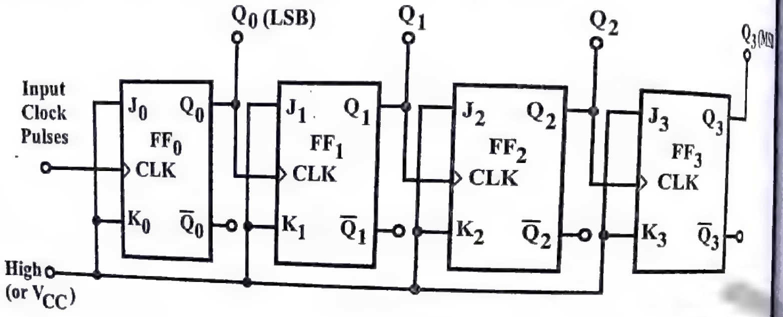


***Fig. 3.33 Logic Diagram of Four-bit Ripple Counter Using J-K Flip-flops***

The first stage is fed by the clock. Noticed that one output pulse is obtained for every two input pulses from the first flip-flop output. Hence, the frequency of the output $Q_0$ is half the frequency of the clock. Hence the first stage can be thought of as a binary counter or as a frequency divider which divided by a factor of 2.

**Working** – The pulses to be counted are applied to the clock input of $FF_0$. The output of $FF_0$ drives $FF_1$, the output of $FF_1$ drives $FF_2$ and the output of $FF_2$ drives $FF_3$. For all stages J and K are tied to the supply voltage, so that J = K = 1. The overall propagation delay time of the counter is the sum of the individual delays of flip-flops.

The clock pulses are applied only to the CLK input of $FF_0$. Thus, $FF_0$ changes to its opposite state, each time the clock pulses make a negative transition state. The normal output of $FF_0$ acts as the clock input for $FF_1$ and so $FF_1$ will toggle each time the $Q_0$ output goes from 1 to 0. In a similar way, $FF_2$ will toggle when $Q_1$ output goes from 1 to 0 and $FF_3$ will toggle when $Q_2$ output goes from 1 to 0. The flip-flop outputs $Q_3$, $Q_2$, $Q_1$ and $Q_0$ represents a 4-bit binary number with $Q_3$ as the MSB. Initially assume that all flip-flops have been cleared to the logic-0 state. The waveforms of the 4-bit ripple counter is shown in fig. 3.34. The truth table of 4-bit binary ripple counter is illustrated in table 3.11.

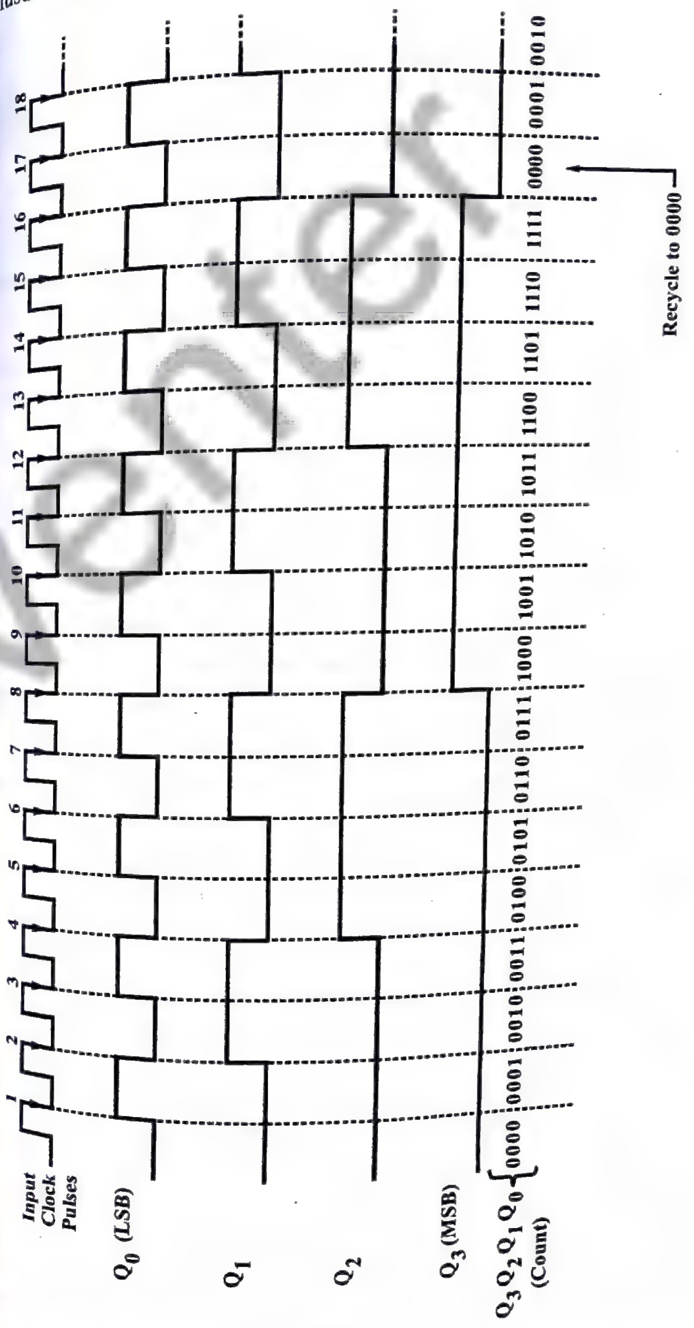


***Fig. 3.34 Waveforms of a 4-bit Ripple Counter***

**Table 3.11 Truth Table of 4-bit Ripple Counter**

| *Number of Input Pulses* | *Flip-flop Outputs* | | | |
|---|---|---|---|---|
| | $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ |
| 0 (Initially) | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 |
| 6 | 0 | 1 | 1 | 0 |
| 7 | 0 | 1 | 1 | 1 |
| 8 | 1 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 |
| 10 | 1 | 0 | 1 | 0 |
| 11 | 1 | 0 | 1 | 1 |
| 12 | 1 | 1 | 0 | 0 |
| 13 | 1 | 1 | 0 | 1 |
| 14 | 1 | 1 | 1 | 0 |
| 15 | 1 | 1 | 1 | 1 |
| 16 (Recycles) | 0 | 0 | 0 | 0 |

***Q.31. How does a counter works as frequency divider ? Explain with suitable example.*** ***(R.G.P.V., Dec. 2016)***

***Ans.*** Refer the ans. of Q.30.

***Q.32. Describe the working of asynchronous 4-bit up-down counter. Also draw its logic diagram.***

***Ans.*** A combination of the up-counter and the down-counter is called the ***up-down counter***. Since the up-down counter has the capability of counting upwards as well as downwards, it is also referred to as multimode counter. In the case of an up-counter, each flip-flop is triggered by normal output of the preceding flip-flop; in the case of a down-counter, each flip-flop is triggered by the inverted output of the preceding flip-flop. In both the counters, the first flip-flop is triggered by the input pulses.

**Working** – A 4-bit up-down counter is shown in fig. 3.35 whose operation is controlled by the up and down control inputs. Table 3.12 shows the counting sequence of up-down counter in the two modes of counting.

Here, to switch the individual stages from count-up to count-down mode, three logic gates per stage are needed. The logic gates are used to permit either the non-inverted output or the inverted output of one flip-flop to the clock input of the following flip-flop, depending on the status of the control inputs. If the count-up line is held at high, while the count-down line is at low, lower AND gates $G_2$, $G_4$ and $G_6$ will be disabled and their outputs are zero. Therefore, it will have no impact on the outputs of OR gates. In addition, the upper AND gates $G_1$, $G_3$ and $G_5$ will be enabled, i.e. it will permit $Q_0$ to pass through the OR gate and into the clock input of the $FF_1$. In the same type, the $Q_1$ output and $Q_2$ output will be gated into the clock input of $FF_2$ and $FF_3$, respectively. Hence, as input pulses are applied, the counter will count up and follow a natural binary counting sequence from 0000 to 1111. With count-up = 0, count-down = 1, the upper AND gates $G_1$, $G_3$ and $G_5$ are disabled and the lower AND gates $G_2$, $G_4$ and $G_6$ are enabled, permitting $\overline{Q}_0$, $\overline{Q}_1$ and $\overline{Q}_2$, to pass through to the clock inputs of the following flip-flops. Therefore, for this condition, the counter will count down as input pulses are applied.

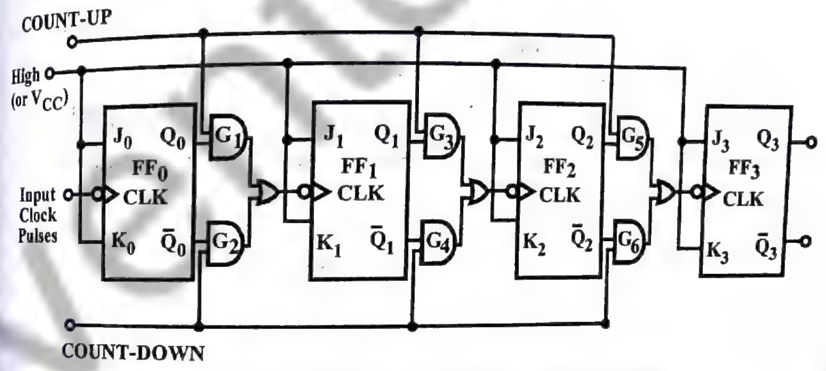


***Fig. 3.35 Asynchronous 4-bit Up-down Counter***

**Table 3.12 Truth Table of 4-bit Up-down Counter**

| *Number of Input Clock Pulses* | *Flip-flop Outputs* | | | | *Number of Input Clock Pulses* | *Flip-flop Outputs* | | | |
|---|---|---|---|---|---|---|---|---|---|
| | *Count-up Mode* | | | | | *Count-down Mode* | | | |
| | $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ | | $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ |
| 0 | 0 | 0 | 0 | 0 | 15 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 14 | 1 | 1 | 1 | 0 |
| 2 | 0 | 0 | 1 | 0 | 13 | 1 | 1 | 0 | 1 |
| 3 | 0 | 0 | 1 | 1 | 12 | 1 | 1 | 0 | 0 |
| 4 | 0 | 1 | 0 | 0 | 11 | 1 | 0 | 1 | 1 |
| 5 | 0 | 1 | 0 | 1 | 10 | 1 | 0 | 1 | 0 |
| 6 | 0 | 1 | 1 | 0 | 9 | 1 | 0 | 0 | 1 |
| 7 | 0 | 1 | 1 | 1 | 8 | 1 | 0 | 0 | 0 |
| 8 | 1 | 0 | 0 | 0 | 7 | 0 | 1 | 1 | 1 |
| 9 | 1 | 0 | 0 | 1 | 6 | 0 | 1 | 1 | 0 |
| 10 | 1 | 0 | 1 | 0 | 5 | 0 | 1 | 0 | 1 |
| 11 | 1 | 0 | 1 | 1 | 4 | 0 | 1 | 0 | 0 |
| 12 | 1 | 1 | 0 | 0 | 3 | 0 | 0 | 1 | 1 |
| 13 | 1 | 1 | 0 | 1 | 2 | 0 | 0 | 1 | 0 |
| 14 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| 15 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 15 | 1 | 1 | 1 | 1 |

If the control inputs are both 0 or 1, then counter will not count-up or count-down since the clock inputs of $FF_1$, $FF_2$ and $FF_3$ will be held constant at either 0 or 1. The $FF_1$ will keep because it is always being clocked. Generally, these conditions are not used.

***Q.33. Describe the BCD ripple counter.*** ***(R.G.P.V., Dec. 2008)***

***Ans.*** The logic diagram of a BCD ripple counter is shown in fig. 3.36. The four outputs are designated by the letter symbol Q with a numeric subscript equal to the binary weight of the corresponding bit in the BCD code. The flip-flops trigger on the negative edge, i.e., when the count pulse (CP) signal goes from 1 to 0. Note that the output of $Q_1$ is applied to the CP inputs of both $Q_2$ and $Q_8$ and the output of $Q_2$ is applied to the CP input of $Q_4$. The J and K inputs are connected either to a permanent 1 signal or to outputs of flip-flops, as shown in the diagram.

A BCD-ripple counter is an asynchronous sequential circuit and cannot be described by Boolean equation developed for describing clocked sequential circuits. Signals that affect the flip-flop transition depend on the order in which they change from 1 to 0. The operation of the counter can be explained by a list of conditions for flip-flop transitions. These conditions are derived from the logic diagram and from knowledge of how a JK-flip-flop operates. Remember that when the CP input goes from 1 to 0, the flip-flop is set if J = 1, is cleared if K = 1, is complemented if J = K = 1, and is left unchanged if J = K = 0. The following are the conditons for each flip-flop state transition –

(i) $Q_1$ is complemented on the negative edge of every count pulse.

(ii) $Q_2$ is complemented if $Q_8 = 0$ and $Q_1$ goes from 1 to 0. $Q_2$ is cleared if $Q_8 = 1$ and $Q_1$ goes from 1 to 0.

(iii) $Q_4$ is complemented when $Q_2$ goes from 1 to 0.

(iv) $Q_8$ is complemented when $Q_4 Q_2 = 11$ and $Q_1$ goes from 1 to 0. $Q_8$ is cleared if either $Q_4$ or $Q_2$ is 0 and $Q_1$ goes from 1 to 0.

To verify the operation of the counter is to derive the timing diagram for each flip-flop from the conditions just listed. This diagram is shown in fig. 3.37 with the binary states after each clock pulse. $Q_1$ changes state after each clock pulse.

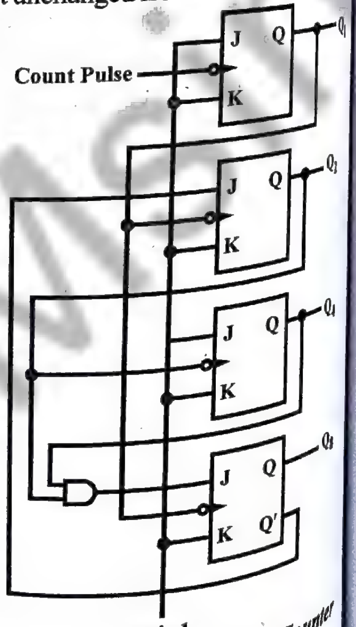


*Fig. 3.36 BCD Ripple Counter*

$Q_2$ complements every time $Q_1$ goes from 1 to 0 as long as $Q_8 = 0$. When $Q_8$ becomes 1, $Q_2$ remains cleared at 0. $Q_4$ complements every time $Q_2$ goes from 1 to 0. $Q_8$ remains cleared as long as $Q_2$ or $Q_4$ is 0, when both $Q_2$ and $Q_4$ because 1's, $Q_8$ complements when $Q_1$ goes from 1 to 0. $Q_8$ is cleared on the next transition of $Q_1$.

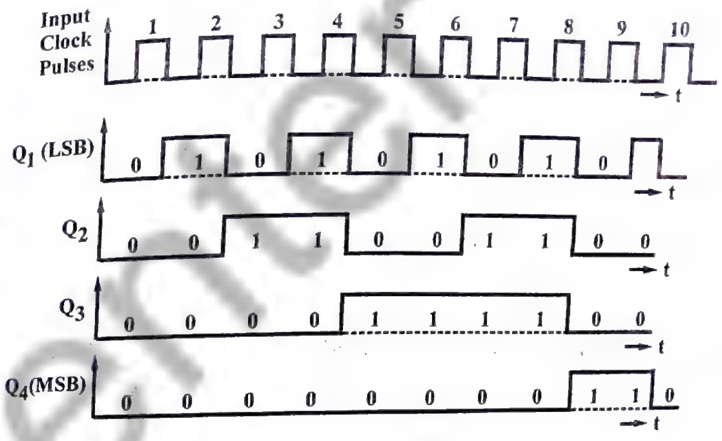


*Fig. 3.37 Timing Diagram of BCD-ripple Counter*

***Q.34. Draw and explain the logic diagram of a 3-bit binary counter.*** ***(R.G.P.V., June 2017)***

***Ans.*** Fig. 3.38 illustrates a 3-bit synchronous binary counter using J-K flip-flops. Here we note that $Q_0$ output changes on each clock pulse as the counter progresses from its original state to its final state and then back to its original state. For operation the $FF_0$ must be held in the toggle mode by the inputs $J_0$ and $K_0$ connecting to high state. The $Q_1$ output goes to the opposite state at $Q_0$ output is a high state. This change occurs at clock pulse-2, clock pulse-4, clock pulse-6 and clock pulse-8. The clock pulse-8 causes the counter to recycle. For this operation, $Q_2$ output is connected to the $J_1$ and $K_1$ inputs of $FF_1$.

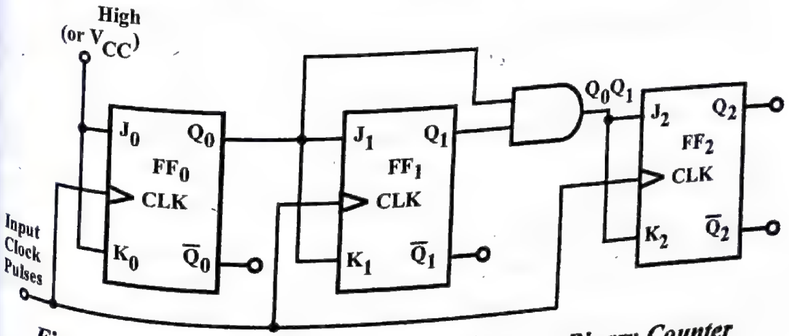


*Fig. 3.38 Logic Diagram of 3-bit Synchronous Binary Counter*

The truth table of 3-bit synchronous binary counter is depicted in table 3.13 and its output waveforms are shown in fig. 3.39.

**Table 3.13 Truth Table of 3-bit Synchronous Counter**

| *Number of Input Clock Pulse* | *Flip-flop Outputs* | | |
|---|---|---|---|
| | $Q_2$ | $Q_1$ | $Q_0$ |
| 0 (Initially) | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 2 | 0 | 1 | 0 |
| 3 | 0 | 1 | 1 |
| 4 | 1 | 0 | 0 |
| 5 | 1 | 0 | 1 |
| 6 | 1 | 1 | 0 |
| 7 | 1 | 1 | 1 |
| 8 (Recycles) | 0 | 0 | 0 |

When $Q_0$ output is a logic-1 and a clock pulse occurs, $FF_1$ is in the no-change mode and therefore changes state. When $Q_0$ output is a logic-0, $FF_1$ is in the no-change mode and remains in its present state. The $Q_2$ output changes state when both $Q_0$ and $Q_1$ outputs are high (or logic-1). This condition is detected by the AND gate and applied to the $J_2$ and $K_2$ inputs of $FF_2$. Whenever both $Q_0$ and $Q_1$ outputs are high, the output of the AND gate makes the $J_2$ and $K_2$ inputs of $FF_2$ high and $FF_2$ toggles on the following clock pulse.

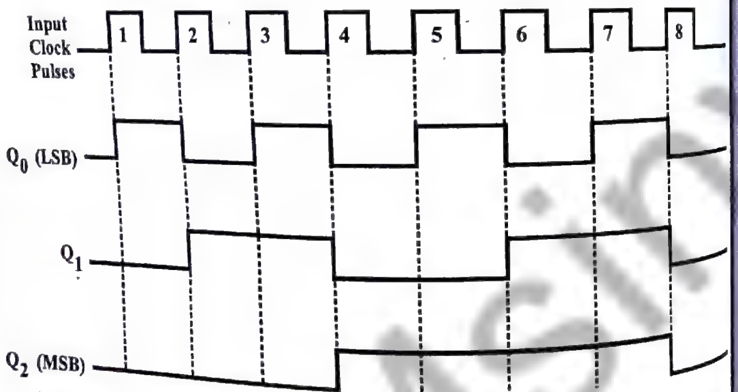


*Fig. 3.39 Output Waveforms of 3-bit Synchronous Binary Counter*

***Q.35. Explain the working of four bit synchronous counter.***

***(R.G.P.V., Dec. 2014)***

***Or***

***Draw and explain 4 bit synchronous binary counter. (R.G.P.V., June 2017)***

***Ans.*** A 4-bit (MOD-16) synchronous counter with parallel carry is illustrated in fig. 3.40. In this counter, the clock inputs of all the flip-flops are connected together, so the input clock signal is applied simultaneously to each flip-flop. The J and K inputs of $FF_0$ is permanently connected to high (or $V_{CC}$) while the J and K inputs of remaining flip-flops are driven by some combination of flip-flop outputs. The $FF_0$ changes its state with the occurrence of negative transition at each clock pulse. The $FF_1$ changes its state when $Q_0 = 1$ and when there is negative transition at clock input. The $FF_2$ changes its state when $Q_0 = Q_1 = 1$ and when there is negative transition at clock input. In similar way $FF_3$ changes state when $Q_0 = Q_1 = Q_2 = 1$ and when there is negative transition at clock input.

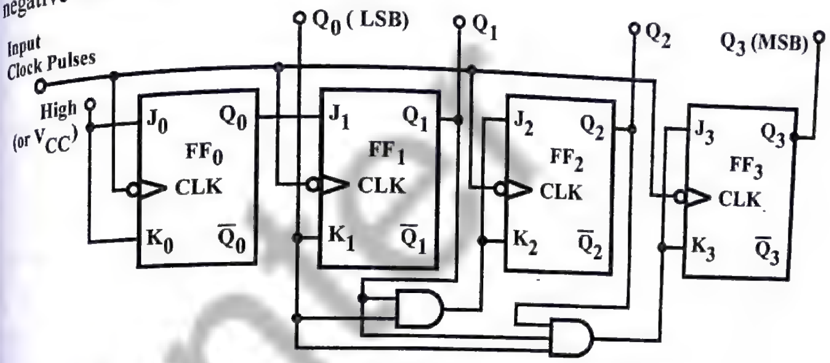


*Fig. 3.40 Four-bit (or MOD-16) Synchronous Counter*

The response time of synchronous counter is defined as the sum of the time taken by one flip-flop to toggle and the time for the new logic levels to propagate through a single AND gate to reach the J and K inputs of all the flip-flops. The speed of operation of synchronous counter is limited only by the propagation delays of a single flip-flop and an AND gate.

The truth table of 4-bit synchronous counter is illustrated in table 3.14.

**Table 3.14 Truth Table of 4-bit Synchronous Counter**

| *Number of Input Clock Pulses* | *Flip-flop Outputs* | | | |
|---|---|---|---|---|
| | $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ |
| 0 (Initially) | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 |
| 6 | 0 | 1 | 1 | 0 |
| 7 | 0 | 1 | 1 | 1 |
| 8 | 1 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 |
| 10 | 1 | 0 | 1 | 0 |
| 11 | 1 | 0 | 1 | 1 |
| 12 | 1 | 1 | 0 | 0 |
| 13 | 1 | 1 | 0 | 1 |
| 14 | 1 | 1 | 1 | 0 |
| 15 | 1 | 1 | 1 | 1 |
| 16 (Recycles) | 0 | 0 | 0 | 0 |

The maximum frequency of operation of the synchronous counter is illustrated as –

$$f_{max} = \frac{1}{t_p + t_g}$$

where, $t_p$ = Propagation delay of one flip-flop
$t_g$ = Propagation delay of one AND gate.

***Q.36. Describe the 4-bit up-counter with the help of logic diagram. Also give its waveforms.***

***Or***

***Design a 4-bit synchronous up counter using J-K flip-flops.***

***(R.G.P.V., Dec. 2015)***

*Ans.* The logic diagram of 4-bit up-counter is shown in fig. 3.41. In a synchronous counter the clock signal is applied to all the flip-flops simultaneously. The truth table of a 4-bit up-counter is depicted in table 3.15 and its waveforms is shown in fig. 3.42. Observing the up-counting sequence, we note that $Q_1$ output changes state for every clock pulse. Therefore, $FF_1$ has to be in toggle mode. The $Q_2$ output changes state whenever $Q_1$ output is logic-1, i.e., $FF_2$ toggles whenever $Q_1$ output is logic-1. So, connect $Q_1$ output to the inputs $J_2$ and $K_2$. The $Q_2$ output changes state whenever $Q_2 = 1$ and $Q_1 = 1$, i.e., $FF_3$ toggles whenever $Q_1 Q_2 = 1$. So, $Q_1 Q_2$ is connected to inputs $J_3$ and $K_3$. The $Q_4$ output changes state whenever $Q_1 = 1$, $Q_2 = 1$ and $Q_3 = 1$, i.e., $FF_4$ toggles when $Q_1 Q_2 Q_3 = 1$. So, $Q_1 Q_2 Q_3$ is connected to the inputs $J_4$ and $K_4$.

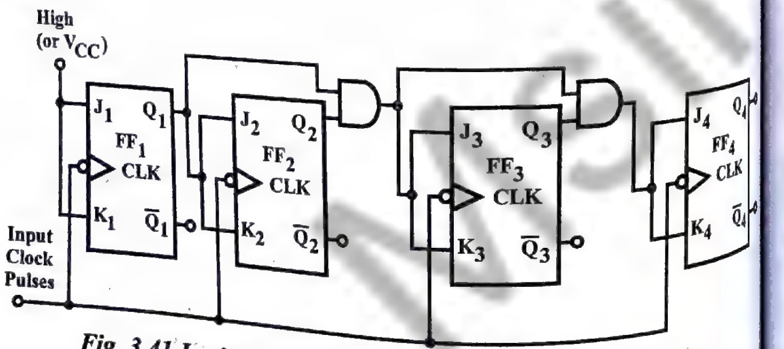


*Fig. 3.41 Logic Diagram of 4-bit Synchronous Up-counter*

**Table 3.15 Truth Table of 4-bit Synchronous Up-counter**

| *Number of Input Clock Pulse* | *Flip-flop Outputs* | | | |
|---|---|---|---|---|
| | $Q_4$ | $Q_3$ | $Q_2$ | $Q_1$ |
| 0 (Initially) | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 |
| 6 | 0 | 1 | 1 | 0 |
| 7 | 0 | 1 | 1 | 1 |
| 8 | 1 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 |
| 10 | 1 | 0 | 1 | 0 |
| 11 | 1 | 0 | 1 | 1 |
| 12 | 1 | 1 | 0 | 0 |
| 13 | 1 | 1 | 0 | 1 |
| 14 | 1 | 1 | 1 | 0 |
| 15 (Recycles) | 1 | 1 | 1 | 1 |
| 16 | 0 | 0 | 0 | 0 |
| 17 | 0 | 0 | 0 | 1 |

It may be observed the waveforms in fig. 3.42, the propagation delay does not get accumulated in synchronous counters as it does in ripple counters.

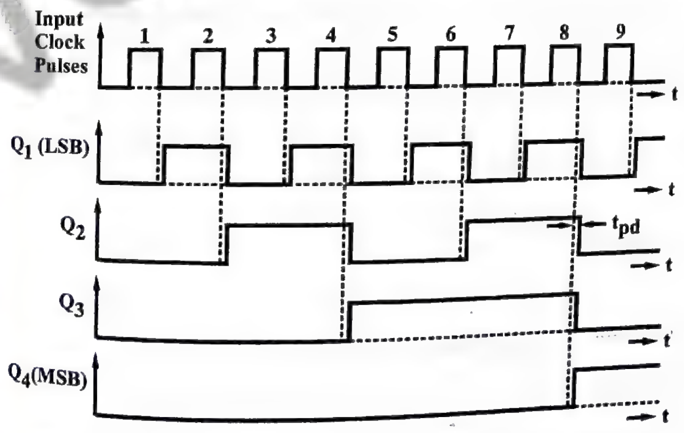


*Fig. 3.42 Waveforms of 4-bit Synchronous Up-counter*

***Q.37. Write short note on synchronous up/down counters.***

***(R.G.P.V., Nov./Dec. 2007, June 2008)***

***Or***

***Design a 4 bit up-down binary counter.*** ***(R.G.P.V., June 2010)***

*Ans.* A synchronous up/down counter can be obtained by combining the up-counting and down-counting operations in a single counter using control or mode signal. Here we discuss about 4-bit synchronous up/down counter. Let us say, we want the counter to count up when mode signal M = 1 and count down when mode signal M = 0. We can obtain the expressions for excitations of an up/down counter by combining the excitations of up and down counter using the mode signal. Therefore, the design equations for an up/down counter are –

$J_1 = K_1 = 1$

$J_2 = K_2 = (Q_1.Up) + (\overline{Q}_1.Down) = Q_1M + \overline{Q}_1\overline{M}$

$J_3 = K_3 = (Q_1.Q_2.Up) + (\overline{Q}_1.\overline{Q}_2.Down) = Q_1Q_2M + \overline{Q}_1\overline{Q}_2\overline{M}$

$J_4 = K_4 = (Q_1.Q_2.Q_3.Up) + (\overline{Q}_1.\overline{Q}_2.\overline{Q}_3.Down) = Q_1Q_2Q_3M + \overline{Q}_1\overline{Q}_2\overline{Q}_3\overline{M}$

The logic diagram of a 4-bit synchronous up/down counter is shown in fig. 3.43. Most up/down counters can be reversed at any point in the sequence.

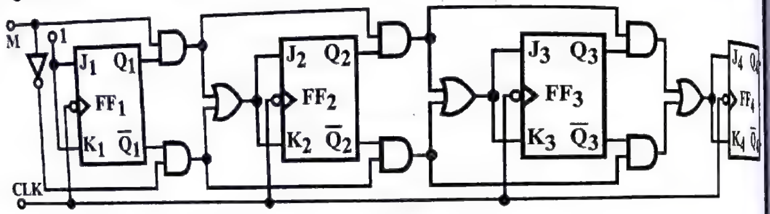


*Fig. 3.43 4-bit Synchronous Up/Down Counter*

***Q.38. What is counter ? Give their types and explain up and down counter.*** ***(R.G.P.V., June 2011)***

***Ans.*** **Counter** – Refer the ans. of Q.26.

**Types** – Counters can be broadly classified under three heads as follows –

(i) Asynchronous and synchronous counters

(ii) Single and multi-mode counters

(iii) Modulus counters.

**Up and Down Counters** – Refer the ans. of Q.37.

***Q.39. Explain the operation of BCD counter.*** ***(R.G.P.V., May/June 2009)***

***Ans.*** A BCD counter counts in binary-coded decimal from 0000 to 1001 and back to 0000. Because of the return to 0 after a count of 9, a BCD counter does not have a regular pattern as in a straight binary count. To derive the circuit of BCD synchronous counter, it is necessary to go through a design procedure.

The excitation table of a BCD counter is given in table 3.16. The excitation for the T flip-flop is obtained from present and next state conditions. An output Y is also shown in the table. This output is equal to 1 when counter present state is 1001. In this way, Y can enable the count of the next-higher order decade while the same pulse switches the present state decade from 1001 to 0000.

**Table 3.16 Excitation Table for BCD Counter**

| *Present State* | | | | *Next State* | | | | *Output* | *Flip-flop Inputs* | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $Q_8$ | $Q_4$ | $Q_2$ | $Q_1$ | $Q_8$ | $Q_4$ | $Q_2$ | $Q_1$ | $y$ | $TQ_8$ | $TQ_4$ | $TQ_2$ | $TQ_1$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |

The flip-flop input function from the excitation table can be simplified by means of maps. The unused state for minterms 10 to 15 are taken as don't care terms. The simplified functions are –

$$TQ_1 = 1$$

$$TQ_2 = Q_8' Q_1$$

$$TQ_4 = Q_2 Q_1$$

$$TQ_8 = Q_8 Q_1 + Q_4 Q_2 Q_1$$

$$y = Q_8 Q_1$$

The circuit can be easily drawn with four T flip-flops, five AND gates, and one OR gate.

Synchronous BCD counter can be cascaded to form a counter for decimal number of any length.

***Q.40. Write short note on ring counter.*** ***[R.G.P.V., June 2002 (CS), 2003, Dec. 2006]***

***Ans.*** This is the simplest shift register counter. This is essentially a circulating shift register connected so that last flip-flop shift its value into the first flip-flop. The basic ring counter using D-type flip-flops is shown in fig. 3.44. If the serial output $Q_0$ of the shift register is connected back to serial input, then an injected pulse will keep circulating. This circuit is referred to as a ring counter. Assuming a starting state of $Q_3 = 1$ and $Q_2 = Q_1 = Q_0 = 0$. After the first clock pulse, the 1 has shifted from output $Q_3$ to output $Q_2$, so that the counter is in the 0100 state. The second clock pulse produces the 0010 state and the third clock pulse produces the 0001 state. On the fourth clock pulse, the 1 from output $Q_0$ is transferred to output $Q_3$, resulting in the 1000 state, i.e. initial state. The subsequent pulses cause the sequence to repeat.

The counter functions as a MOD-4 counter, since it has four distinct states before the sequence repeats. Although this circuit does not progress through the normal binary counting sequence, it is still a counter because each count corresponds to a unique set of flip-flop states. We note that each flip-flop output waveform has a frequency equal to one-forth of the clock frequency since this is a MOD-4 ring counter.

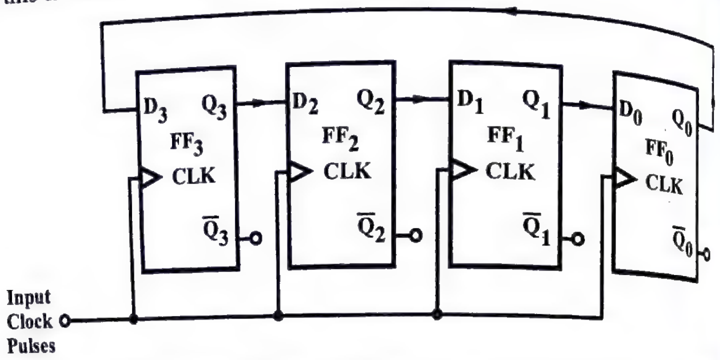


*Fig. 3.44 Logic Diagram of Four-bit Ring Counter using D-type Flip-flops*

A ring counter is still useful because it can be decoded without the use of decoding gates. The decoding signal for each state is obtained at the output of its corresponding flip-flop. The waveforms for 4-bit ring counter is illustrated in fig. 3.45. This counter circuit can also be used for counting the number of pulses. The number of pulses counted is read by noting which flip-flop is in 1 state.

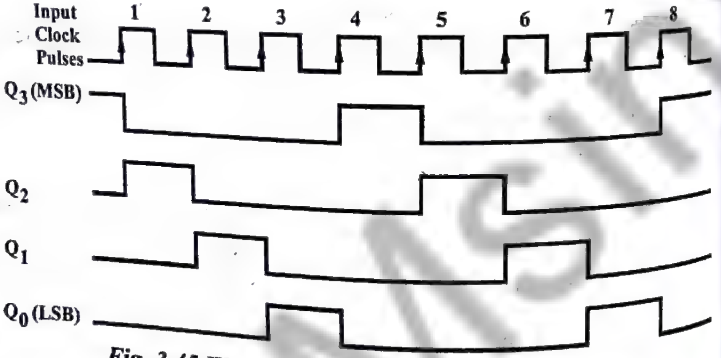


*Fig. 3.45 Waveforms of Four-bit Ring Counter*

***Q.41. Using D-flip-flops construct a ring counter. Discuss its operation and applications in digital systems.*** ***(R.G.P.V., Dec. 2011)***

***Ans.*** Refer the ans. of Q.40.

***Q.42. Draw a diagram for a 5-bit ring counter using J-K flip-flop and explain its working.*** ***(R.G.P.V., Nov./Dec. 2007, June 2009)***

***Ans.*** Ring counter is the simplest shift register counter. The basic ring counter using J-K FFs is shown in fig. 3.46. Its state diagram and the sequence table are shown in fig. 3.47.

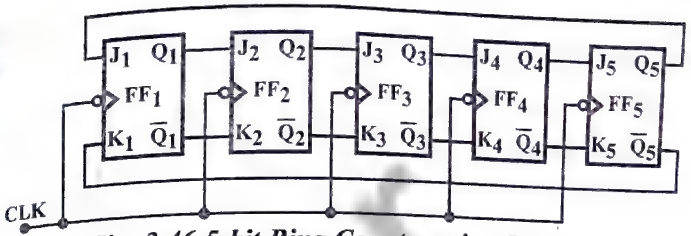
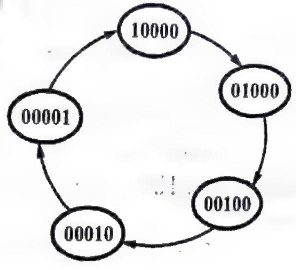


*Fig. 3.46 5-bit Ring Counter using J-K FFs*

In most instances, only a single 1 is in the register and is made to calculate around the register as long as clock pulses are applied.

| $Q_1$ | $Q_2$ | $Q_3$ | $Q_4$ | $Q_5$ | After Clock Pulse |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 2 |
| 0 | 0 | 0 | 1 | 0 | 3 |
| 0 | 0 | 0 | 0 | 1 | 4 |
| 1 | 0 | 0 | 0 | 0 | 5 |
| 0 | 1 | 0 | 0 | 0 | 6 |
| 0 | 0 | 1 | 0 | 0 | 7 |
| 0 | 0 | 0 | 1 | 0 | 8 |
| 0 | 0 | 0 | 0 | 1 | 9 |

***Sequence Table***



***State Diagram***

*Fig. 3.47*

Initially, the first FF is preset to a 1. So, the initial state is 1000, i.e., $Q_1 = 1$, $Q_2 = 0$, $Q_3 = 0$ and $Q_4 = 0$. After each clock pulse, the contents of the register are shifted to the right by one bit and $Q_4$ is shifted to $Q_1$. The sequence repeats after four clock pulses. The number of distinct states in the ring counter, i.e., the mod of the ring counter is equal to the number of FFs used in the counter. An n-bit ring counter can count only n bits whereas an n-bit ripple counter can count $2^n$ bits. So, the ring counter is uneconomical compared to a ripple counter, but has the advantage of requiring no decoder, since we can read the count by simply nothing which FF is set.

***Q.43. What is switch-tail counter ? Give its applications.*** ***(R.G.P.V., Dec. 2004)***

***Ans.*** A modification of the ring counter which uses flip-flops more economically is the switch-tail counter. It is also known as ***Johnson counter.*** A switch-tail counter is shown in fig. 3.48. Here the interconnection between flip-flops is that at the tail the connection has been switched to $\overline{Q}_3$ output. Since the flip-flop array forms a ring, there is really no end to the cascade and the switch can be made between any two flip-flops.

Initially assume that all the flip-flops of the counter have been cleared, so that $Q_0 = Q_1 = Q_2 = Q_3 = 0$. Then the next clock-triggering edge will transfer zeros into $FF_1$, $FF_2$ and $FF_3$, leaving them unaltered. However, because of the switch-in connections at the output of $FF_3$, the $FF_0$ will go to the set state.

Thus the counter goes from $Q_0\,Q_1\,Q_2\,Q_3 = 0000$ to $Q_0\,Q_1\,Q_2\,Q_3 = 1000$. The sequence of states through which the counter gates is given in table 3.17.

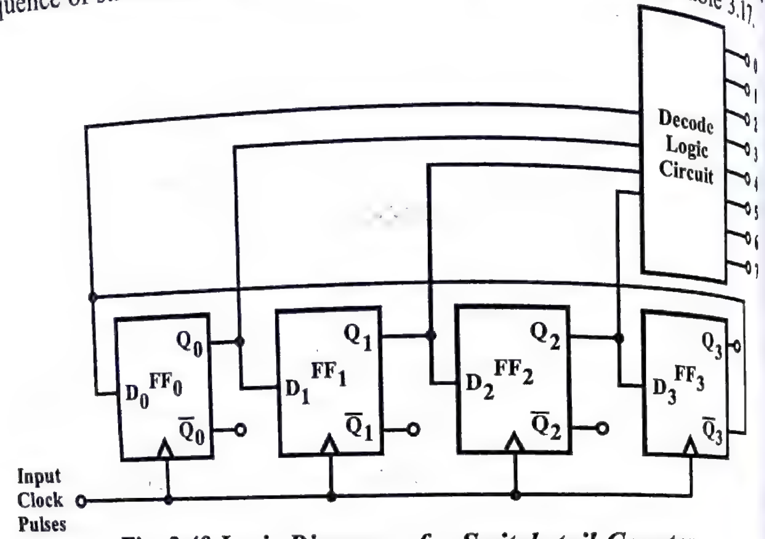


*Fig. 3.48 Logic Diagram of a Switch-tail Counter*

**Table 3.17 Truth Table of Switch-tail Counter**

| *Count Sequence* | *Flip-flop Outputs* | | | | *Decode Logic* |
|---|---|---|---|---|---|
| | $Q_0$ | $Q_1$ | $Q_2$ | $Q_3$ | |
| 0 | 0 | 0 | 0 | 0 | $\overline{Q_0}\,\overline{Q_3}$ |
| 1 | 1 | 0 | 0 | 0 | $\overline{Q_0}\,\overline{Q_1}$ |
| 2 | 1 | 1 | 0 | 0 | $Q_1\,\overline{Q_2}$ |
| 3 | 1 | 1 | 1 | 0 | $Q_2\,\overline{Q_3}$ |
| 4 | 1 | 1 | 1 | 1 | $Q_0\,Q_3$ |
| 5 | 0 | 1 | 1 | 1 | $\overline{Q_0}\,Q_1$ |
| 6 | 0 | 0 | 1 | 1 | $\overline{Q_1}\,Q_2$ |
| 7 | 0 | 0 | 0 | 1 | $\overline{Q_2}\,Q_3$ |
| 0 | 0 | 0 | 0 | 0 | $\overline{Q_0}\,\overline{Q_3}$ |

From the truth table in table 3.17 we observe that the counter cycles through eight states, i.e., twice the number of flip-flops and then returns to its initial state. Thus with 'n' flip-flops the ring counter has a modulo-n while the switch-tail counter has a modulo 2n.

At each count, one output is singled out by being at logic level-1 while all others are at logic-0. It is more economical than the normal ring counter but less economical than the ripple counter.

**Applications** – The counters are used in a wide range of applications like counting of any variable – time, speed, distance, frequency generation and division, waveform generation, sequence generation, repeating displays, etc.

***Q.44. Design a 4-bit Johnson counter.*** ***(R.G.P.V., Dec. 2010)***

***Ans.*** Refer the ans. of Q.43.

## NUMERICAL PROBLEMS

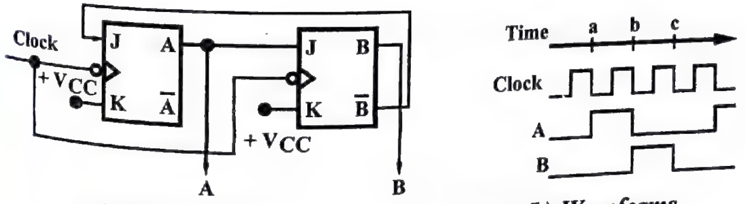
***Prob.1. How many flip-flops are required to construct a MOD-128 counter? A MOD-32 ? What is the largest decimal number that can be stored in a MOD-64 counter ?*** ***(R.G.P.V., Dec. 2010)***

***Sol.*** A MOD-128 counter must have seven flip-flops, since $2^7 = 128$. Five flip-flops are needed to construct a MOD-32 counter. The largest decimal number that can be stored in a six flip-flop counter (MOD-64) is 111111 = 63.

***Prob.2. What modulus counters can be constructed with the use of four flip-flops ?*** ***(R.G.P.V., Dec. 2010)***

***Sol.*** A four flip-flop counter has a natural count of 16. We can thus construct any counter that has a modulus between 16 and 2, inclusive. We might choose to use four flip-flops only for counters having a modulus between 16 and 9, since only three flip-flops are required for a modulus of less than 8 and only two are required for a modulus of less than 4.

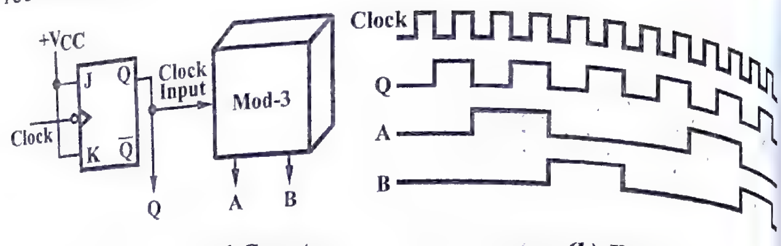
***Prob.3. Draw the waveform expected from the MOD-6 counter by connecting a single flip-flop in front of MOD-3 counters in figure given below.***



*(a) Logic Diagram* *(b) Waveforms*

***Fig. 3.49*** ***(R.G.P.V., Dec. 2010)***

***Sol.*** The resulting counter is a 2 × 3 = MOD-6 counter that has the waveforms as shown in fig. 3.50. But B now has a period equal to six clock periods, but it is not symmetrical.



(a) 2 × 3 MOD-6 Counter (b) Waveforms

Fig. 3.50

*Prob.4. A certain JK flip-flop has a $t_p = 12$ ns. What is the largest MOD counter that can be constructed using flip-flops and still operate upto 10 MHz.* **(R.G.P.V., Dec. 2011)**

***Sol.*** For a state change to ripple through all n stages

$$T_{clock} = nt_{pd}$$

where n = Number of flip-flops.

Therefore, the maximum frequency is given by

$$f_{max} = \frac{1}{T_{clock}} = \frac{1}{nt_{pd}}$$

$$\therefore \quad n = \frac{1}{t_{pd} \times f_{max}}$$

$$= \frac{1}{12\times10^{-9}\times10\times10^{6}} = 8.33$$

Here, we take n = 8, then

MOD-number = $2^n = 2^8 =$ **256** Ans.

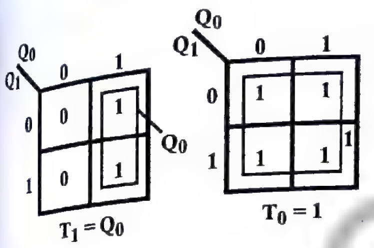
***Prob.5. Design a MOD-4 counter using T flip flop.*** **(R.G.P.V., Dec. 2012)**

***Sol.*** There are 4 states in a MOD-4 counter, which requires two flip-flops. The excitation table for synchronous MOD-4 counter using T-flip flops is shown in table 3.18.

**Table 3.18 Excitation Table**

| Present State | | Next State | | Required Excitation | |
|---|---|---|---|---|---|
| $Q_1$ | $Q_0$ | $Q_1$ | $Q_0$ | $T_1$ | $T_0$ |
| 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 |

The K-maps for $T_1$ and $T_0$ inputs in terms of $Q_1$ and $Q_0$ are shown in fig. 3.51.



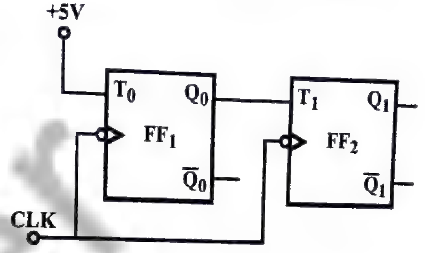


Fig. 3.51 Fig. 3.52

The schematic logic diagram is shown in fig. 3.52.

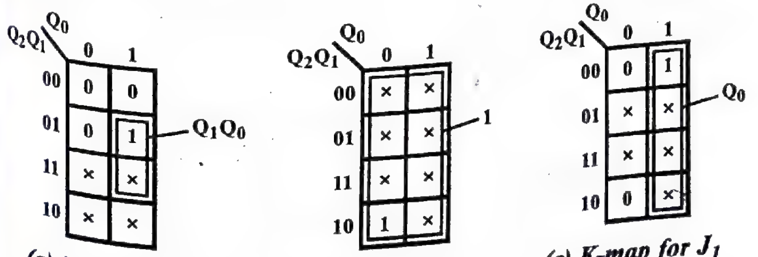
***Prob.6. Design a MOD-5 counter.*** **(R.G.P.V., May 2018)**

***Sol.*** The counting sequence of Mod-5 counter is 000, 001, 010, 011, 100 and 000. It needs three flip-flops and have three invalid states 101, 110, 111. The entries for excitation corresponding to invalid states are don't cares. The excitation table for Mod-5 counter using J-K flip-flop is shown in table 3.19.

**Table 3.19 Excitation Table For Mod-5 Counter**

| Present State (PS) | | | Next State (NS) | | | Excitation Inputs | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $Q_2$ | $Q_1$ | $Q_0$ | $Q_2$ | $Q_1$ | $Q_0$ | $J_2$ | $K_2$ | $J_1$ | $K_1$ | $J_0$ | $K_0$ |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | × | 0 | × | 1 | × |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | × | 1 | × | × | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | × | × | 0 | 1 | × |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | × | × | 1 | × | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | × | 1 | 0 | × | 0 | × |

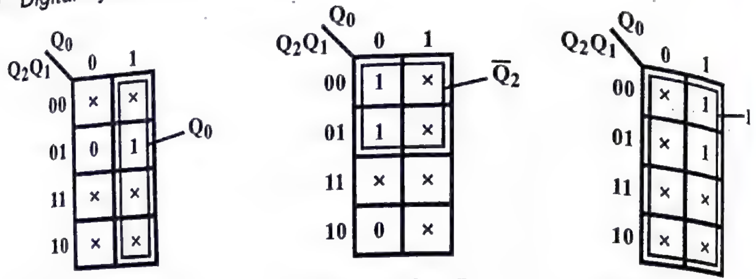
The K-maps for $J_2$, $K_2$, $J_1$, $K_1$, $J_0$, $K_0$ based on the excitation table 3.19 are given in fig. 3.53.



(a) K-map for $J_2$
$J_2 = Q_1Q_0$

(b) K-map for $K_1$
$K_2 = 1$

(c) K-map for $J_1$
$J_1 = Q_0$



*(d) K-map for $K_1$*

$K_1 = Q_0$

*(e) K-map for $J_1$*

$J_0 = \overline{Q}_2$

*(f) K-map for $K_0$*

$K_0 = 1$

**Fig. 3.53**

The logic diagram of the Mod-5 counter using J-K flip-flops is shown in fig. 3.54.

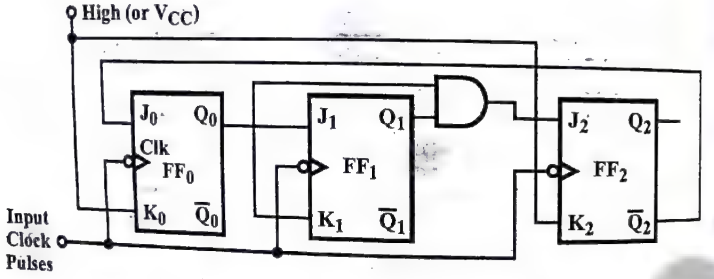


*Fig. 3.54 Logic Diagram of MOD-5 Counter*

***Prob.7. Design a synchronous BCD-up counter using minimum number of J-K flip-flops and gates.*** ***(R.G.P.V., June 2005)***

***Or***

***Design a decade counter.*** ***(R.G.P.V., June 2009)***

***Or***

***Design a synchronous BCD counter with JK flip-flops.*** ***(R.G.P.V., Dec. 2010)***

***Sol.*** Decade counter follows a sequence of ten states and returns to 0 after the count of 9. Such a counter must have at least four flip-flop to represent each decimal digit, since a decimal digit is represented by a binary code with at least four bits. The sequence of states in a decimal counter is dictated by the binary code used to represent a decimal digit. If BCD is used, the sequence is as shown in the state diagram of the fig. 3.55.

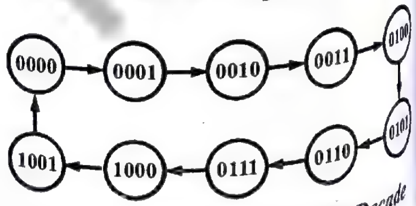


*Fig. 3.55 Logic Diagram of Decade Counter*

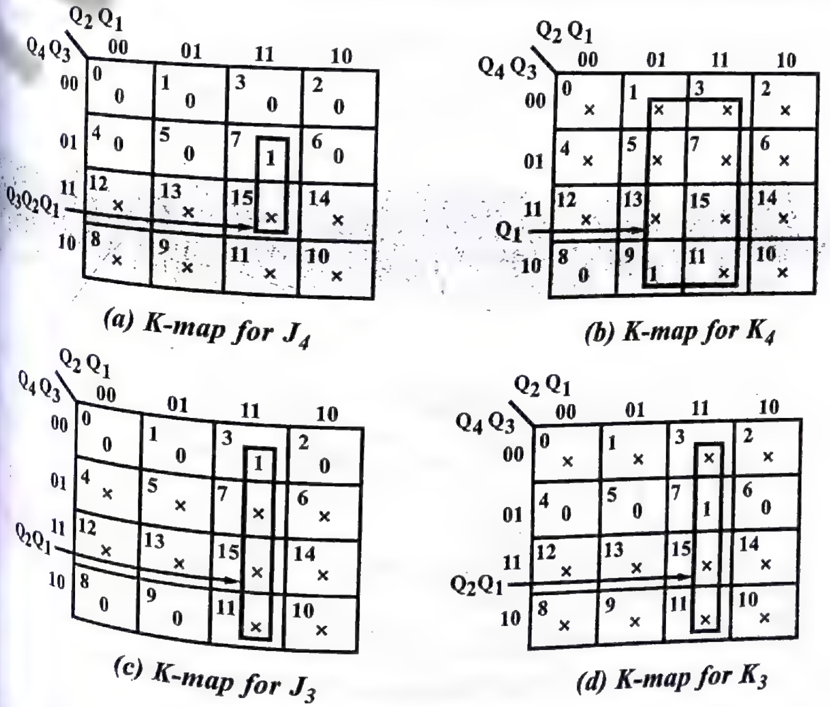
Decade counter counts from 0 to 9. To count in decimal from 0 to 99, we need two decade counter. To count from 0 to 999, we need three decade counter. Multiple decade counter can be constructed by connecting decade counter in cascade, one for each decade.

**Table 3.20 Table for Count Sequence and Flip-flop Inputs**

| *Present State or Counter State* | | | | *Flip-flop Inputs* | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $Q_4$ | $Q_3$ | $Q_2$ | $Q_1$ | $J_4$ | $K_4$ | $J_3$ | $K_3$ | $J_2$ | $K_2$ | $J_1$ | $K_1$ |
| 0 | 0 | 0 | 0 | 0 | × | 0 | × | 0 | × | 1 | × |
| 0 | 0 | 0 | 1 | 0 | × | 0 | × | 1 | × | × | 1 |
| 0 | 0 | 1 | 0 | 0 | × | 0 | × | × | 0 | 1 | × |
| 0 | 0 | 1 | 1 | 0 | × | 1 | × | × | 1 | × | 1 |
| 0 | 1 | 0 | 0 | 0 | × | × | 0 | 0 | × | 1 | × |
| 0 | 1 | 0 | 1 | 0 | × | × | 0 | 1 | × | × | 1 |
| 0 | 1 | 1 | 0 | 0 | × | × | 0 | × | 0 | 1 | × |
| 0 | 1 | 1 | 1 | 1 | × | × | 1 | × | 1 | × | 1 |
| 1 | 0 | 0 | 0 | × | 0 | 0 | × | 0 | × | 1 | × |
| 1 | 0 | 0 | 1 | × | 1 | 0 | × | 0 | × | × | 1 |

The BCD counter is nothing but a MOD-10 counter or a decade counter, which needs four flip-flops. The remaining six states are unused states.The count sequence and the flip-flop inputs given are in table 3.20.

The K-maps for $J_4$, $K_4$, $J_3$, $K_3$, $J_2$, $K_2$, $J_1$ and $K_1$ are given in fig. 3.56.

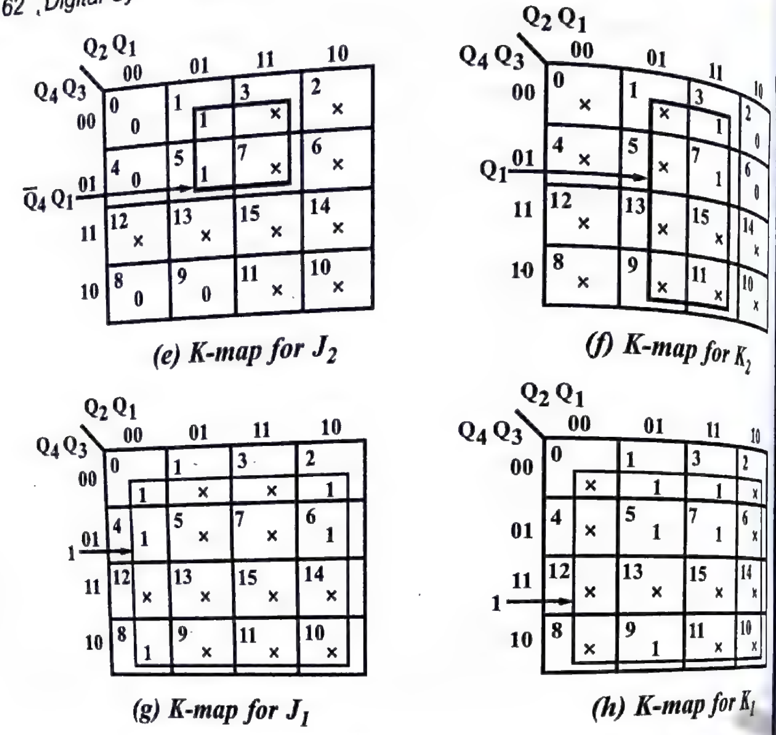


*(a) K-map for $J_4$*

*(b) K-map for $K_4$*

*(c) K-map for $J_3$*

*(d) K-map for $K_3$*



(e) K-map for $J_2$

(f) K-map for $K_2$

(g) K-map for $J_1$

(h) K-map for $K_1$

*Fig. 3.56 K-maps for Excitations of the Decade-up Counter using J-K Flip-flops*

The minimized expressions obtained from the K-maps in fig. 3.56 are-

$J_4 = Q_3 Q_2 Q_1$, $K_4 = Q_1$ ...(i)

$J_3 = Q_2 Q_1$, $K_3 = Q_2 Q_1$ ...(ii)

$J_2 = \overline{Q}_4 Q_1$, $K_2 = Q_1$ ...(iii)

$J_1 = 1$, $K_1 = 1$ ...(iv)

The implementation of decade-up counter using above expressions is illustrated in fig. 3.57.

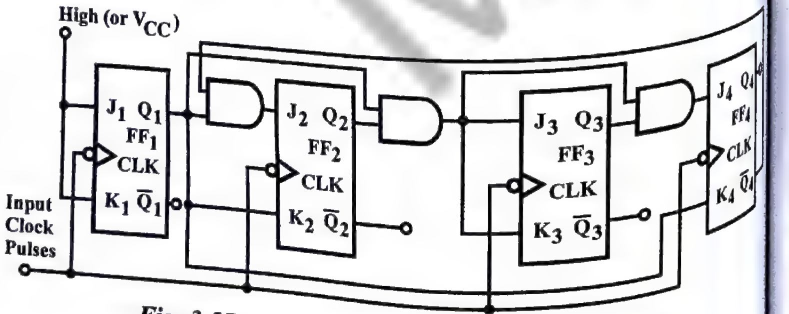


*Fig. 3.57 Logic Diagram of Decade-up Counter*

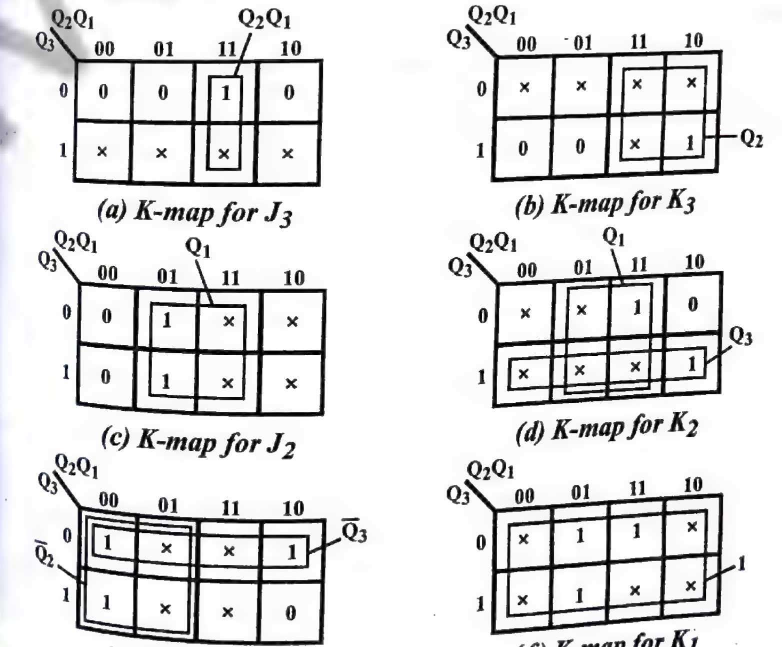
**Prob.8. Design a MOD-7 counter using J-K flip-flop. (R.G.P.V., June 2012)**

*Sol.* The counting sequence for a MOD-7 counter is 000, 001, 010, 011, 100, 101, 110, and 000. It has one invalid state 111 and needs three flip-flops. The entires for excitations corresponding to invalid states are don't cares. The excitation table for synchronous MOD-7 counter using J-K flip-flops is shown in table 3.21.

**Table 3.21 Excitation Table for MOD-7 Counter using J-K Flip-flops**

| *Present State* | | | *Next State* | | | *Required Excitation* | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $Q_3$ | $Q_2$ | $Q_1$ | $Q_3$ | $Q_2$ | $Q_1$ | $J_3$ | $K_3$ | $J_2$ | $K_2$ | $J_1$ | $K_1$ |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | × | 0 | × | 1 | × |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | × | 1 | × | × | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | × | × | 0 | 1 | × |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | × | × | 1 | × | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | × | 0 | 0 | × | 1 | × |
| 1 | 0 | 1 | 1 | 1 | 0 | × | 0 | 1 | × | × | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 | × | 1 | × | 1 | 0 | × |

The K-maps for $J_3$, $K_3$, $J_2$, $K_2$, $J_1$ and $K_1$ in terms of $Q_3$, $Q_2$ and $Q_1$ are shown in fig. 3.58.



(a) K-map for $J_3$

(b) K-map for $K_3$

(c) K-map for $J_2$

(d) K-map for $K_2$

(e) K-map for $J_1$

(f) K-map for $K_1$

*Fig. 3.58 K-maps for Excitations of Synchronous MOD-7 Counter using J-K Flip-flops*

The minimized expressions obtained from the K-maps are as follows –

$J_3 = Q_2Q_1$ $\qquad K_3 = Q_2$

$J_2 = Q_1$ $\qquad K_2 = Q_1 + Q_3$

$J_1 = \overline{Q}_2 + \overline{Q}_3$ $\qquad K_1 = 1$

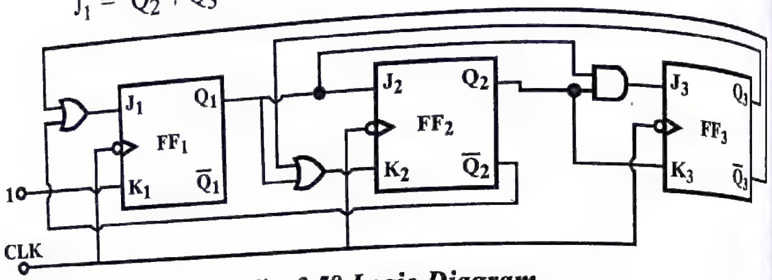


*Fig. 3.59 Logic Diagram*

**Prob. 9. Design a MOD-6 counter using J-K flip flops.**

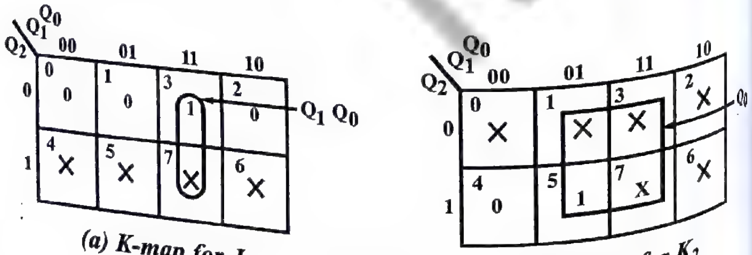
*(R.G.P.V., Dec. 2012)*

*Sol.* The counting sequence for MOD-6 counter is 000,001,010,011,100,101 and 000. It needs three flip-flops and has two invalid states, 110 and 111. The entries for excitations corresponding to invalid states are don't cares. The excitation table for synchronous MOD-6 counter using J-K flip-flops is shown in table 3.22.

**Table 3.22 Excitation Table for MOD-6 Counter using J-K Flip-flops**

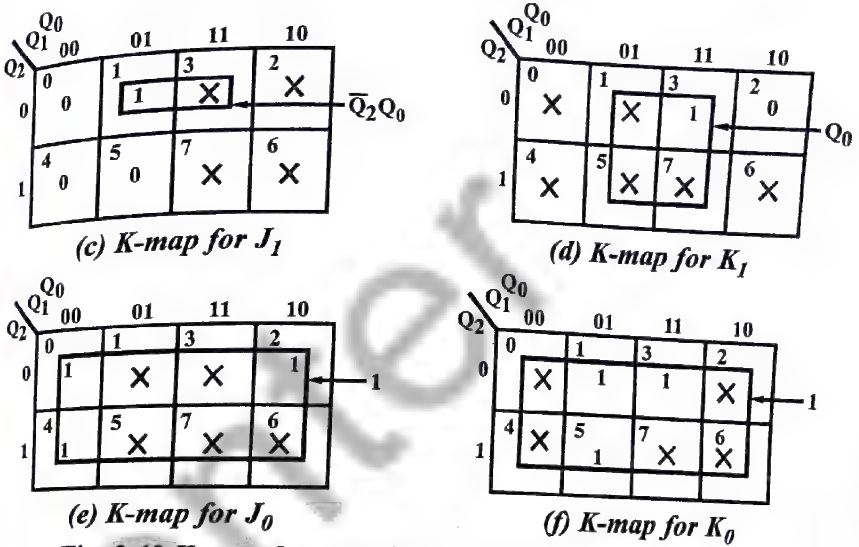
| *Present State (PS)* | | | *Next State (NS)* | | | *Excitation Inputs* | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $Q_2$ | $Q_1$ | $Q_0$ | $Q_2$ | $Q_1$ | $Q_0$ | $J_2$ | $K_2$ | $J_1$ | $K_1$ | $J_0$ | $K_0$ |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | × | 0 | × | 1 | × |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | × | 1 | × | × | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | × | × | 0 | 1 | × |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | × | × | 1 | × | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | × | 0 | 0 | × | 1 | × |
| 1 | 0 | 1 | 0 | 0 | 0 | × | 1 | 0 | × | × | 1 |

The K-maps for $J_2$, $K_2$, $J_1$, $K_1$, $J_0$ and $K_0$ in terms of $Q_2$, $Q_1$ and $Q_0$ is shown in fig. 3.60.



*(a) K-map for $J_2$*

*(b) K-map for $K_2$*



*(c) K-map for $J_1$*

*(d) K-map for $K_1$*

*(e) K-map for $J_0$*

*(f) K-map for $K_0$*

***Fig. 3.60 K-map for Excitations of Synchronous MOD-6 Counter Using J-K Flip-flops***

The minimal expressions for excitations obtained from the K-maps are illustrated as follows –

$$J_2 = Q_1 Q_0 \qquad \text{...(i)}$$

$$K_2 = Q_0 \qquad \text{...(ii)}$$

$$J_1 = \overline{Q}_2 Q_0 \qquad \text{...(iii)}$$

$$K_1 = Q_0 \qquad \text{...(iv)}$$

$$J_0 = 1 \qquad \text{...(v)}$$

$$K_0 = 1 \qquad \text{...(vi)}$$

The logic diagram of the synchronous MOD-6 counter using J-K flip-flops is illustrated in fig. 3.61.

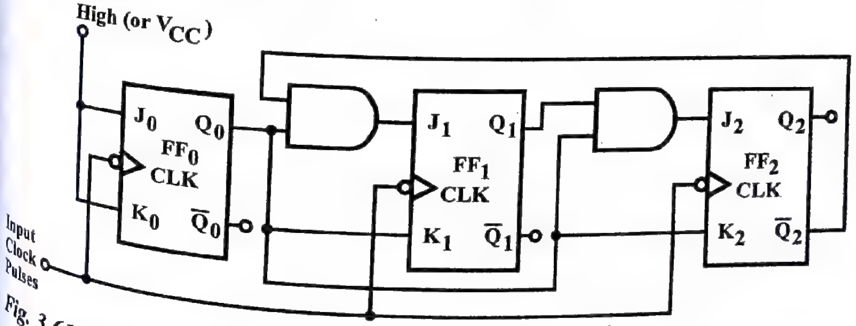


***Fig. 3.61 Logic Diagram of the Synchronous MOD-6 Counter using J-K Flip-flops***

**Prob.10. Design a MOD-12 binary counter using J-K flip-flop.**
**(R.G.P.V., June 2015)**

*Sol.* For designing of MOD-12 counter the number of flip-flops (n) such that MOD no. ≤ $2^n$. Therefore here mod number is 12, for satisfying this condition n should be 4 i.e. $12 \leq 2^4$, Therefore here four flip-flops are used.

The counting sequence of MOD-12 counter has 12 states i.e. 0 to 11. Let us assume that the MOD-12 counter has 12 states viz. a, b, c, d, e, f, g, h, i, j, k, and *l*.

**Step-I State Diagram** – Now the state diagram for the MOD-12 counter can be drawn as shown in fig. 3.62.

Here, it is assumed that the state transition from one state to another state takes place when the clock pulse is asserted. When the clock is unasserted, the counter remains in the present state.

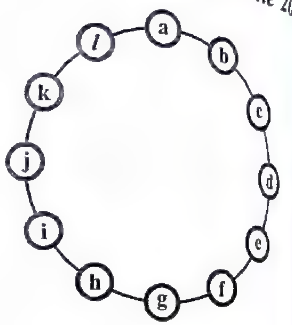


*Fig. 3.62 State Diagram of MOD-12 Counter*

**Step II State Table** – From the above state diagram, one can draw the PS-NS table as shown in table 3.23.

**Table 3.23 PS-NS Table for MOD-12 Counter**

| *Present State (PS)* | *Next State (NS)* |
|---|---|
| a | b |
| b | c |
| c | d |
| d | e |
| e | f |
| f | g |
| g | h |
| h | i |
| i | j |
| j | k |
| k | *l* |
| *l* | a |

The above state table does not have any redundant state because no two states are equivalent. So, there is no modification required in the above state stable.

**Step-III State Assignment** – Let us assign four state variables to these states a, b, c, d, e, f, g, h, i, j, k and *l* as follows : a = 0000, b = 0001, c = 0010, d = 0011, e = 0100, f = 0101, g = 0110, h = 0111, i = 1000, j = 1001, k = 1010, and *l* = 1011. Then the above PS-NS table can be modified as shown in table 3.24.

**Table 3.24 PS-NS Table for MOD-12**

| *Present State (PS)* | | | | *Next State (NS)* | | | |
|---|---|---|---|---|---|---|---|
| $q_3$ | $q_2$ | $q_1$ | $q_0$ | $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | × | × | × | × |
| 1 | 1 | 0 | 1 | × | × | × | × |
| 1 | 1 | 1 | 0 | × | × | × | × |
| 1 | 1 | 1 | 1 | × | × | × | × |

**Step-IV Excitation Table** – The excitation table having entries for flip-flop inputs ($J_3K_3$, $J_2K_2$, $J_1K_1$ and $J_0K_0$) can be drawn, from the above PS-NS table using the application table of respective flip-flop given in table 3.25.

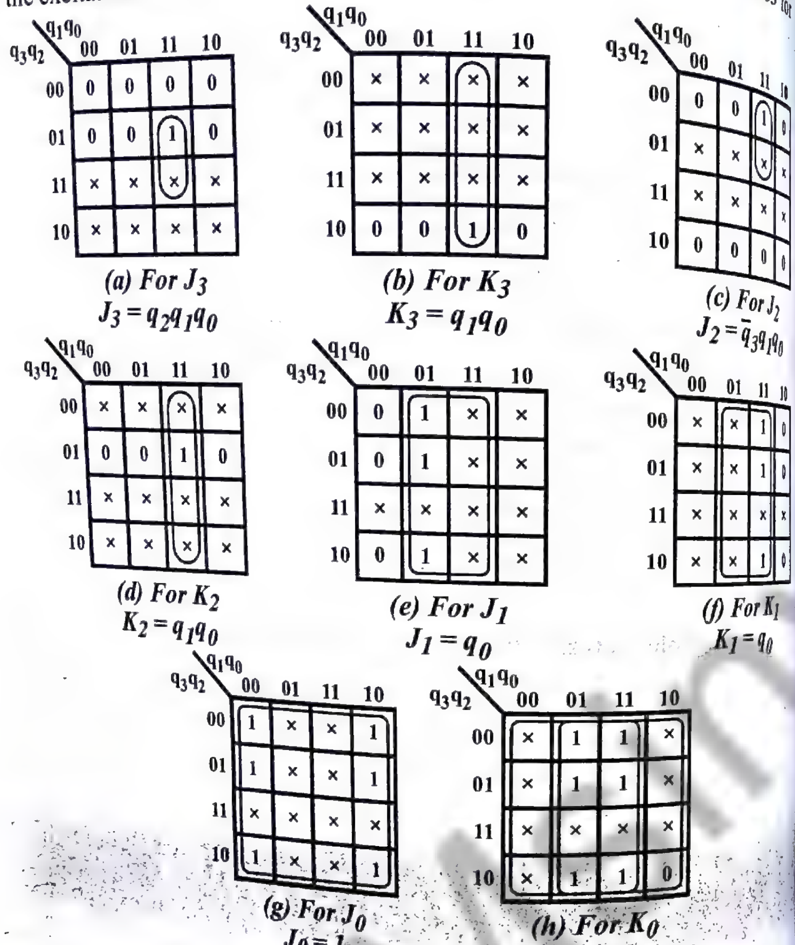
**Table 3.25 Flip-flop Excitation Tables**

| *Q(t)* | *Q(t + 1)* | *J* | *K* |
|---|---|---|---|
| 0 | 0 | 0 | × |
| 0 | 1 | 1 | × |
| 1 | 0 | × | 1 |
| 1 | 1 | × | 0 |

**Table 3.26 Excitation Table for MOD-12 Counter**

| *Present State (PS)* | | | | *Next State (NS)* | | | | *Excitation Input* | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $q_3$ | $q_2$ | $q_1$ | $q_0$ | $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ | $J_3K_3$ | $J_2K_2$ | $J_1K_1$ | $J_0K_0$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 × | 0 × | 0 × | 1 × |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 × | 0 × | 1 × | × 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 × | 0 × | × 0 | 1 × |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 × | 1 × | × 1 | × 1 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 × | × 0 | 0 × | 1 × |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 × | × 0 | 1 × | × 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 × | × 0 | × 0 | 1 × |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 × | × 1 | × 1 | × 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | × 0 | 0 × | 0 × | 1 × |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | × 0 | 0 × | 1 × | × 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | × 0 | 0 × | × 0 | 1 × |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | × 1 | 0 × | × 1 | × 1 |
| 1 | 1 | 0 | 0 | × | × | × | × | × × | × × | × × | × × |
| 1 | 1 | 0 | 1 | × | × | × | × | × × | × × | × × | × × |
| 1 | 1 | 1 | 0 | × | × | × | × | × × | × × | × × | × × |
| 1 | 1 | 1 | 1 | × | × | × | × | × × | × × | × × | × × |

**Step-V Excitation Maps** – The excitation maps for $J_3$, $K_3$, $J_2$, $K_2$, $J_1$, $K_1$ and $J_0$ and $K_0$ inputs of the counter can be drawn as shown in fig. 3.63 for the excitation table.



(a) For $J_3$
$J_3 = q_2q_1q_0$

(b) For $K_3$
$K_3 = q_1q_0$

(c) For $J_2$
$J_2 = \bar{q}_3q_1q_0$

(d) For $K_2$
$K_2 = q_1q_0$

(e) For $J_1$
$J_1 = q_0$

(f) For $K_1$
$K_1 = q_0$

(g) For $J_0$
$J_0 = 1$

(h) For $K_0$
$K_0 = 1$

*Fig. 3.63 Excitation Maps for MOD-12 Counter*

**Step-VI Schematic Diagram** – Using the above excitation equations, the circuit diagram for the mod-12 counter can be drawn as shown in fig. 3.64.

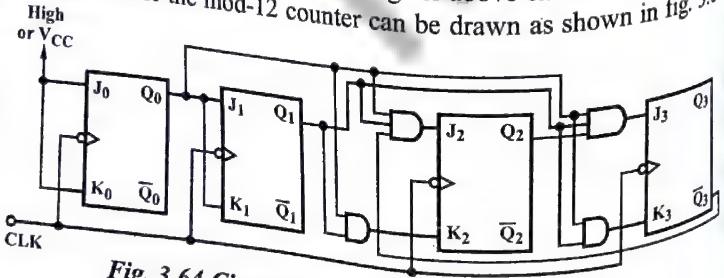


*Fig. 3.64 Circuit Diagram for MOD-12 Counter*

***Prob.11. Find the MOD number of counter in fig. 3.65. Determine its counting sequence. Draw the state diagram. Find the frequency at output $Q_D$ if input frequency is 7 kHz.***

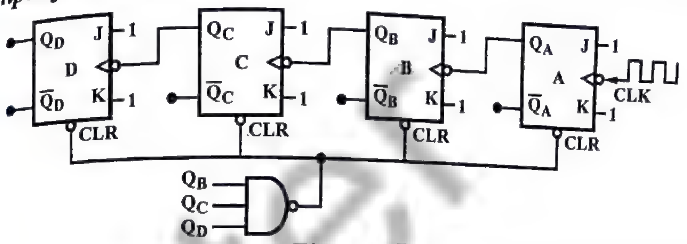


*Fig. 3.65* **(R.G.P.V., June 2014)**

***Sol.*** The basic ripple counter of fig. 3.65 is limited to a MOD-number that is equal to $2^n$, where n is the number of flip-flops.

MOD number = $2^n = 2^4 = 16$ **Ans.**

With the NAND gate, this counter functions as follows –

(i) The NAND gate output is connected to the CLEAR inputs of each flip-flop. As long as the NAND gate output is HIGH, it will have no effect on the counter. When the NAND gate output goes LOW, it will clear all flip-flops, and the counter immediately goes to the 0000 state.

(ii) The outputs of the counter $Q_B$, $Q_C$ and $Q_D$ are given as inputs to the NAND gate. The NAND gate output goes LOW whenever $Q_B = Q_C = Q_D = 1$. This condition will occur when counter goes from the 1110 state to the 1111 state (on the fifteenth input pulse). The LOW at the NAND gate output will clear the counter to the 0000 state. Once the flip-flops have been cleared, the NAND gate output goes back to HIGH, since $Q_B = Q_C = Q_D = 1$ condition no longer exists.

(iii) Although, the counting sequence is 0000 → 0001 → 0010 → 0011 → 0100 → 0101 → 0110 → 0111 → 1000 → 1001 → 1010 → 1011 → 1100 → 1101 → 1110 → 1111 → 0000.

Although the counter does go to the 1111 state, it remains there only for a few nano seconds before it recycles to 0000.

Now, the state diagram for the mod-16 counter can be drawn as shown in fig. 3.66. Here, the state transition from one state to another takes place when the clock pulse is asserted and when the clock is unasserted, the counter remains in the present state.

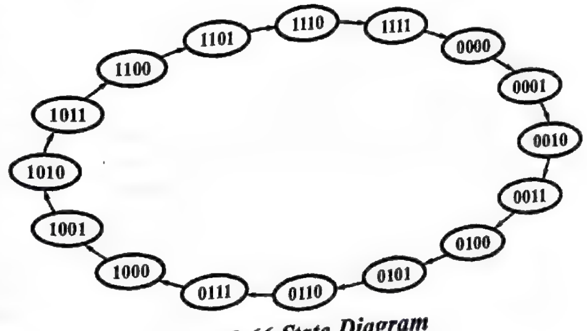


*Fig. 3.66 State Diagram*

Now, the frequency at the output of last stage is

$$f_4 = \frac{f_{in}}{2^4} = \frac{7\text{ kHz}}{16} = \frac{7 \times 10^3}{16} = 437.5\text{ Hz}$$

## SEMICONDUCTOR MEMORIES, INTRODUCTION TO DIGITAL ICs 2716, 2732, etc. & THEIR ADDRESS DECODING, MODERN TRENDS IN SEMICONDUCTOR MEMORIES SUCH AS DRAM, FLASH RAM, etc., DESIGNING WITH ROM AND

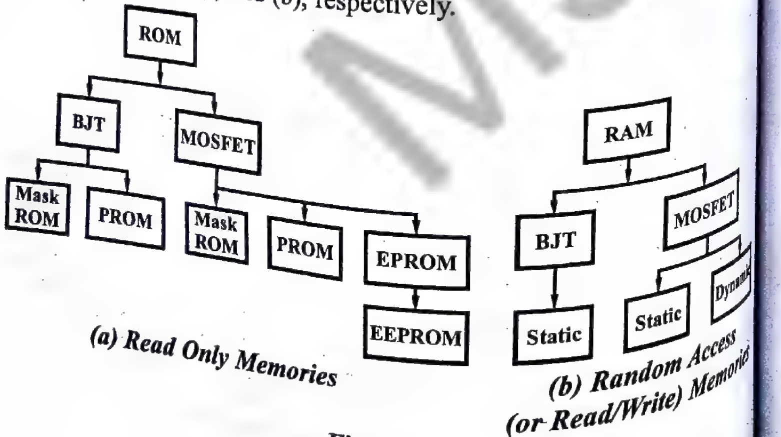
***Q.45. Give a brief introduction of a semiconductor memories.***
***(R.G.P.V., Dec. 2005, 2006, 2008, 2017)***

***Or***

***Give a comparison of various semiconductor memories. (R.G.P.V., June 2014)***

*Ans.* Typical semiconductor memory contains a rectangular array of memory cells, fabricated on a silicon wafer and kept in a convenient package, such as a DIP. A transistor flip-flop is the basic memory cell and it is used to store 1-bit of information. According to the type of transistor used to construct the individual memory cells, memories are classified as bipolar, metal oxide semiconductor (MOS) and complementary metal oxide semiconductor (CMOS). The capacity of memory is determined by the total number of cells, it contains. For example, a 1024 bipolar memory chip is a semiconductor memory having 1024 memory cells. Each cell consists of a flip-flop constructed with the use of bipolar transistors.

A bipolar memory chip has a faster operation but MOS and CMOS memory chips have greater packing density, reduced size and cost and lower power requirements. Generally, memories can be divided into the following categories – random-access or read/write memories (RAM) and read only memories (ROM). The read-only memories (ROM) and RAM memories are shown in figs. 3.67 (a) and (b), respectively.



*(a) Read Only Memories*

*(b) Random Access (or Read/Write) Memories*

***Fig. 3.67***

A RAM is used where data changes frequently. The logic circuitry associated with a RAM will permit a single bit of information to be stored in any of the memory cells, which is the write operation. There is also a logic circuitry to detect whether 0 or 1 is stored in any particular cell, which is the read operation. RAM suggests whether a bit can be written (stored) in any cell or read (detected) from any cell. WRITE ENABLE control signal defines the mode of operation (read or write). In the write mode, information at the data input is written into the selected cell. In the read mode, data is made available at the output from the selected memory. Since each cell is a flip-flop, a loss of power shows a loss of data.

A ROM is used where data does not change. The content of a ROM is fixed at the time of manufacturing, by the oxide-layer thickness or by the presence or absence of a working transistor in a memory cell, or by opening or shorting of the gate structure, or by metallization.

***Q.46. Give broad classification of semiconductor memories.***
***(R.G.P.V., June 2015)***

*Ans.* Refer the ans. of Q.45.

***Q.47. Write short note on read only memory (ROM).***
***(R.G.P.V., June 2005, 2012, Dec. 2016)***

*Ans.* The read only memory (ROM) is a type of semiconductor memory, which is designed to hold data that either are permanent or will not change frequently. During normal operation, no new data can be written into a ROM, but data can be read from ROM. Some ROMs provide the facility of storing data, which must be built-in during the manufacturing process. In case of other ROMs, the data can be entered electrically. The process of entering data is known as the programming in the ROM.

A block diagram of ROM is shown in fig. 3.68. It consists of n-input lines and m-output lines. Each bit combination of the input variables is known as address. Each bit combination that comes out of the output is called a word. The number of bits per word is equal to the number of m-output lines. An address is essentially a binary number that denotes the minterms of n-variables. The number of distinct addresses possible with n-input variables is $2^n$. An output word can be selected by a unique address. The word available on the output lines at any given time depends

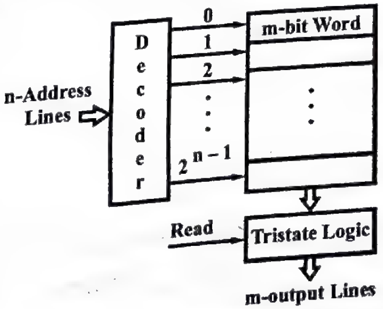


***Fig. 3.68 Block Diagram of ROM***

on the address value applied to the input lines. A ROM is characterized by the number of words $2^n$ and the number of bits per word m.

For example, consider 32 × 8 ROM. The unit consists of 32 words of 8 bits each. Thus, there are eight output lines and there are 32 distinct words stored in the unit, each of which may be applied to the output lines. There are only five inputs in a 32 × 8 ROM because $2^5 = 32$ and with five variables, 32 addresses or minterms can be specified.

ROM is a combinational circuit with AND gates connected as a decoder and a number of OR gates equal to the number of outputs in the unit. The internal logic construction of a 32 × 4 ROM is shown in fig. 3.69. The five input variables are decoded into 32 lines by means of 32 AND gates and 5 inverters. Each output of the decoder shows one of the minterms of a function of five variables. Each one of the 32 addresses chooses one and only one output from the decoder. The address in a 5-bit number given to the inputs and the selected minterm out of the decoder is the one marked with the equivalent decimal number. The 32 outputs of the decoder are connected through fuses to each OR gate.

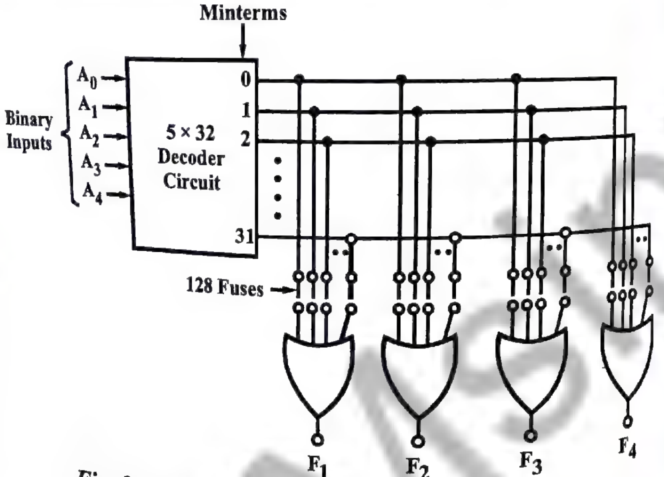


*Fig. 3.69 Logic Construction of a 32 × 4 ROM*

ROM is a two-level implementation in sum of minterms form and it does not have to be an AND-OR implementation, but it can be any other possible two-level minterm implementation. Generally, the second level is a wired-logic connection to provide the blowing of fuses. ROMs are used in the design of digital computer systems and also used in implementing combinational circuits.

***Q.48. What is mask programmed ROM ?***

***Ans.*** The mask programmed ROM contains its storage locations written into (programmed) by the manufacturer according to the customer's specifications. Mask is a photographic negative which is used to control the electrical interconnections on the chip. A special mask is required for each different set of information to be stored in the ROM. Because these masks are costly, so this type of ROM is economical only if a very large quantity of the same ROM is required. Some ROMs of this type are available as off-the-shelf devices programmed with commonly used information or data like mathematical tables and character generator codes for CRT displays. The main drawback of this type of ROM is that it cannot be reprogrammed in the event of a design change requiring a modification of the stored data.

***Q.49. Explain the following –***
***(i) Programmable ROM (PROM)***
***(ii) Erasable Programmable ROM (EPROM)***
***(iii) Electrically Erasable PROM (EEPROM).***

***Ans. (i) PROM*** – The programmable ROMs are used where small quantities are required because it is more economical than the mask-programmed ROM. When ordered, PROM units contain all 0's or all 1's in every bit of the stored words. In the PROM, the fuses are blown by application of current pulses through the output terminals. A blown fuse represents one binary state and an unbroken link shows the other state. This permits the user to program the unit in the laboratory to obtain the desired relationship between input addresses and stored words. Special units called PROM programmers are available commercially to facilitate this procedure.

***(ii) EPROM*** – An EPROM can be erased and it is a nonvolatile memory which will hold its stored data indefinitely. In the process of programming, an EPROM involves the use of special voltage levels to the appropriate chip inputs for a specified amount of time. Generally, a special programming circuit is used in the programming process which is separate from the circuit in which the EPROM will eventually be working. The complete programming process can take up to several minutes for one EPROM chip.

***(iii) EEPROM*** – The electrically erasable PROM is an improvement over the EPROM. This retains the same floating-gate structure as the EPROM, but with the addition of a very thin oxide region above the drain of the MOSFET memory cell. This modification causes the EEPROM's major characteristic– its electrical erasability. A charge can be induced onto the floating gate by applying a high voltage between the MOSFET's gate and drain, where it will remain, even when power is removed; reversal of the same voltage results a removal of the trapped charges from the floating gate and erases the cell.

The EEPROM has another advantage over the EPROM that is its ability to erase and rewrite individual bytes in the memory array electrically.

***Q.50. State and differentiate between ROM, PROM, EPROM and EEPROM.*** ***(R.G.P.V., Dec. 2014)***

***Ans.*** Refer the ans. of Q.47 and Q.49.

### *Q.51. What do you mean by digital IC ?*

***Ans.*** Digital ICs are those circuits, which perform logic functions with the help of binary numbers 0 and 1; like logic gates, flip-flops, counters, shift registers etc. Digital ICs are most popular in realization of electronic systems in the areas of instrumentation, communication, controls and computers. On the basis of technology involved in their manufacturing, digital ICs can be classified into two categories –

(i) Bipolar (ii) MOS families

NPN transistor is the most important device in the bipolar digital ICs and P- and N-channel MOSFETs with complementary symmetry circuitary are popular in MOS digital ICs.

### *Q.52. What are the features of 2716 and 2732 digital ICs. ?*

***Ans.*** The popular EPROM used in 8088 microprocessor system are 2708 (1K × 8), 2716 (2K × 8), 2732 (4K × 8) and 2764 (8K × 4). The features of 2716 are as follows –

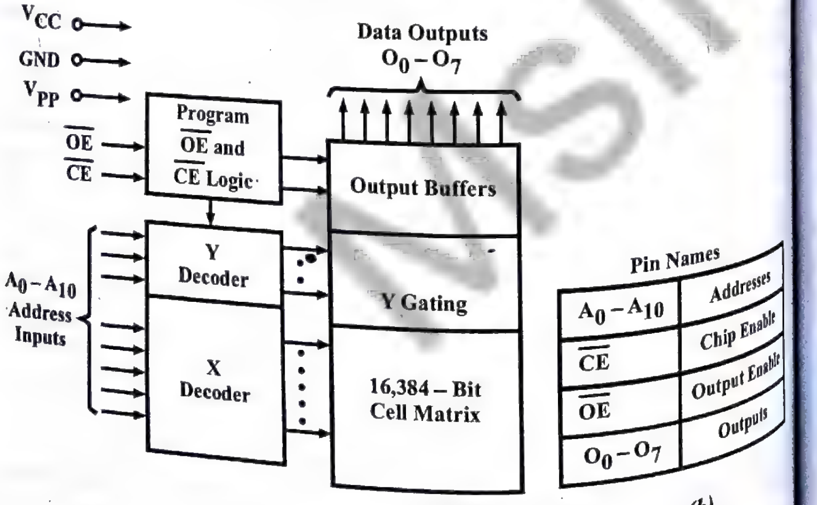
(i) Capacity is 16K. (ii) Access time is 450 ns.
(iii) Number of pins are 24. (iv) Peak voltage is 25 V.

The features of 2732 are as follows –

(i) Capacity of 2732 is 32K. (ii) Access time is 450 ns.
(iii) Number of pins are 24. (iv) Peak voltage is 25 V.

### *Q.53. Discuss Intel 2716 IC in brief.*

***Ans.*** The Intel 2716 is a 16,384-bit ultraviolet erasable and electrically programmable read-only memory (EPROM). The 2716 operates from a single 5-volt power supply, has a static standby mode, and features fast single-address programming. It makes designing with EPROMs fast, easy and economical.



| Pin Names | |
|---|---|
| $A_0 - A_{10}$ | Addresses |
| $\overline{CE}$ | Chip Enable |
| $\overline{OE}$ | Output Enable |
| $O_0 - O_7$ | Outputs |

*(a) Block Diagram*

*(b)*

*Fig. 3.70*

The 2716, with its single 5-volt supply and with an access time upto 350 ns, is ideal for use with high performance + 5 V microprocessors such as Intel's 8085 and 8086. Selected 2716-5s and 2716-6s are also available for slower speed applications. The 2716 also has a static standby mode which reduces power consumption without increasing access time. The maximum active power dissipation is 525 mW while the maximum standby power dissipation is only 132 mW, a 75% savings.

The 2716 uses a simple and fast method for programming – a single TTL-level pulse. There is no need for high voltage pulsing because all programming controls are handled by TTL signals. Programming of any location at any time – either individually, sequentially or at random is possible with the 2716's single-address programming. Total programming time for all 16,384 bits is only 100 seconds.

### *Q.54. What is address decoding ? Discuss it types.*

***Ans.*** Address decoding refers to the way a computer system decodes the addresses on the address bus to select memory locations in one or more memory or peripheral devices. The 68000's 23 bit address bus allows $2^{23}$ 16 bit words to be uniquely addressed.

***(i) Full Address Decoding*** – Each addressable memory location corresponds to a unique address value on the address bus in full address decoding. Full address decoding of two memory devices is shown in fig. 3.71. Memory $M_1$ is selected whenever $A_{12}$-$A_{23}$ = 000000000000, while $M_2$ is selected whenever $A_{12}$-$A_{23}$ = 100000000000.

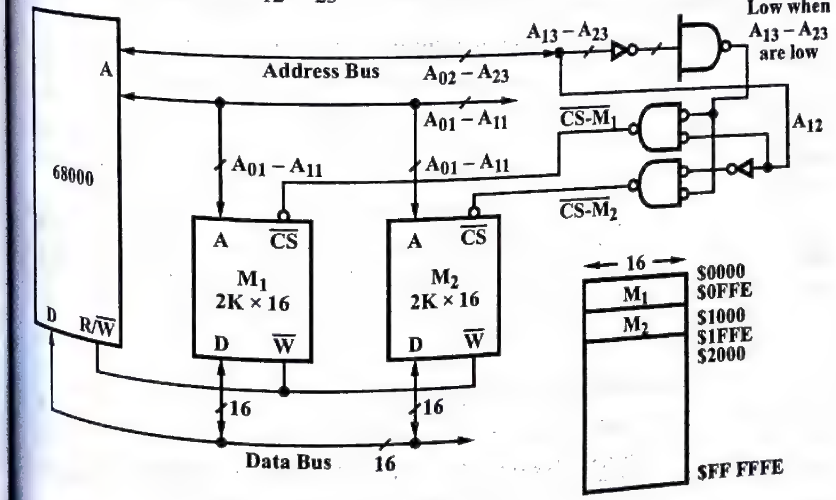


*Fig. 3.71*

***(ii) Partial Address Decoding*** – In partial address decoding, not all address lines in the address bus are used in the decoding process. Partial address decoding of two memory devices is shown in fig. 3.72, where $A_{23}$ is used to distinguish between the two. For this case, $M_1$ and $M_2$ are repeated 2048 times through the memory space. $M_1$ is selected when $A_{23} = 0$ and $M_2$ is selected when $A_{23} = 1$.

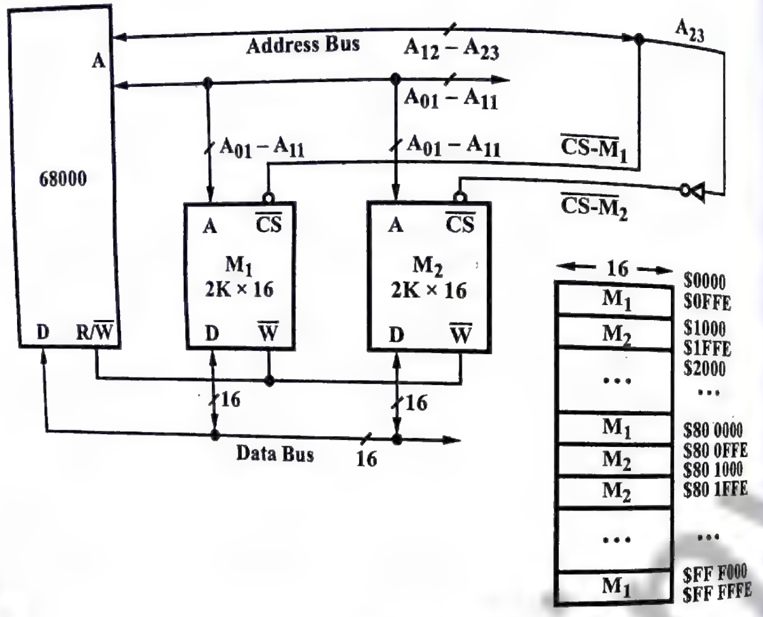


*Fig. 3.72*

***Q.55. Discuss the RAM and ROM.*** ***(R.G.P.V., June 2011)***

***Ans.*** **RAM** – Most of the main memory in a general-purpose computer is made up of RAM integrated circuit chips. Originally, RAM was used to refer to a random-access memory, but now it is used to designate a read/write memory to distinguish it from a read-only memory. RAM is used for storing the bulk of programs and data that are subjected to change. Since the RAM is volatile, its contents are destroyed when power is turned off.

A RAM chip is better suited for communication with the CPU, if it has one or more control inputs that select the chip only when needed. RAM is much faster and more expensive than sequential access.

**ROM** – Refer the ans. of Q.47.

***Q.56. Discuss the static RAM memory.***

***Ans.*** A static RAM essentially contains an array of flip-flops, one of each stored bit. The data written into a flip-flop remains stored as a D.C. power is maintained. The memory capacity of static RAM varies from 64-bits to 1-M bit.

Static RAMs are available in bipolar, MOS and BiCMOS technologies; the majority of applications use NMOS or CMOS RAMs. The bipolars have the advantage in speed and MOS devices have much greater capacities and lower power consumption.

Fig. 3.73 shows the logic diagram of a static RAM cell. The cell is selected by HIGH values on the ROW and COLUMN lines. The input data bit (1 or 0) is written into the cell by setting the flip-flop for a 1 and resetting the flip-flop for a 0 when the READ/$\overline{\text{WRITE}}$ line is LOW. The flip-flop is unaffected, when the READ/$\overline{\text{WRITE}}$ line is HIGH. It means that the stored bit is gated to the data out line.

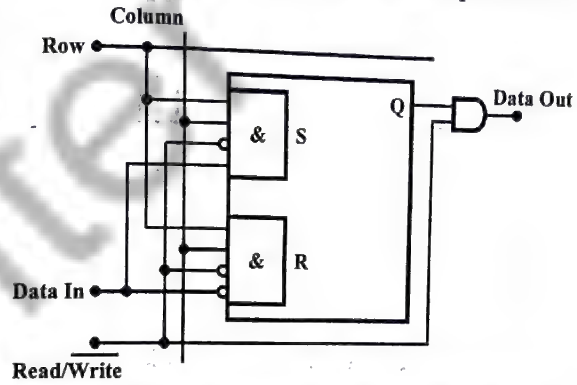


***Fig. 3.73 Logic Diagram of a Static RAM Cell***

***Q.57. Write short note on dynamic RAM memory.***

***Ans.*** Dynamic RAMs are fabricated using MOS technology. Main characteristics of these memories are high capacity, low power requirement and moderate operating speed. Unlike static RAMs, which store information in flip-flops, dynamic RAMs store 1s and 0s as charges on a small MOS capacitor. These charges have the tendency to leak off after a period of time, therefore dynamic RAMs require periodic recharging of the memory cells. This is known as refreshing the dynamic RAM.

The organization of a dynamic memory allows many cells to be accessed by a minimum amount of circuitry. Each memory cell is gated onto a data I/O line when a row and a column select MOSFETs are driven ON as shown in Fig. 3.74.

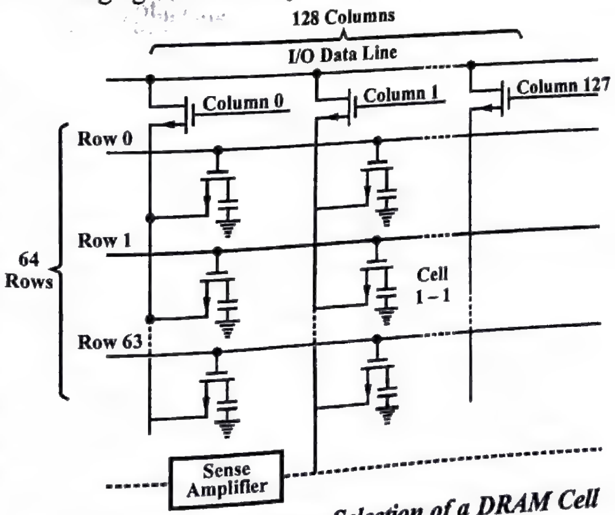


***Fig. 3.74 Row and Column Selection of a DRAM Cell***

**Q.58. Explain the difference between static and dynamic memories.**

***Ans.*** In static memory, the content does not change with time; in dynamic memory, its content changes with time. Dynamic memory cells use the capacitance of a transistor as the storage device. Only one transistor is needed to store one bit of information. The capacitor must be refreshed periodically without being discharged in order to prevent loss of information. Static memory devices require no refreshing, and hold data as long as D.C. power is applied.

**Q.59. Draw the logic diagram of a 4 × 4 RAM and describe the operations.** ***[R.G.P.V., Dec. 2003(CS)]***

***Ans.*** The logic diagram of a 4 × 4 RAM is shown in fig. 3.75. It consists of a 4 words of 4 bits each and has a total of 16 binary cells. Each block labeled BC represents the binary cell with its three inputs and one output. A memory with four words requires two address lines. The two address inputs go through a 2 × 4 decoder to select one of the four words. The decoder is enabled with the memory-enable input. When memory-enable is 0, all outputs of the decoder are 0 and none of the memory words are selected.

With the memory-enable at 1, one of the four words is selected, dictated by the value in the two address lines. Once a word has been selected, the read/write input determines the operation. During the read operation, the four bits of the selected word go through OR gates to the output terminals. During the write operation, the data available in the input lines are transferred into the four binary cells of the selected word. The binary cells which are not selected are disabled and their previous binary values remain unchanged. When the memory-enable input goes into the decoder is equal to 0, none of the words are selected and the contents of all cells remain unchanged regardless of the value of the read/write input.

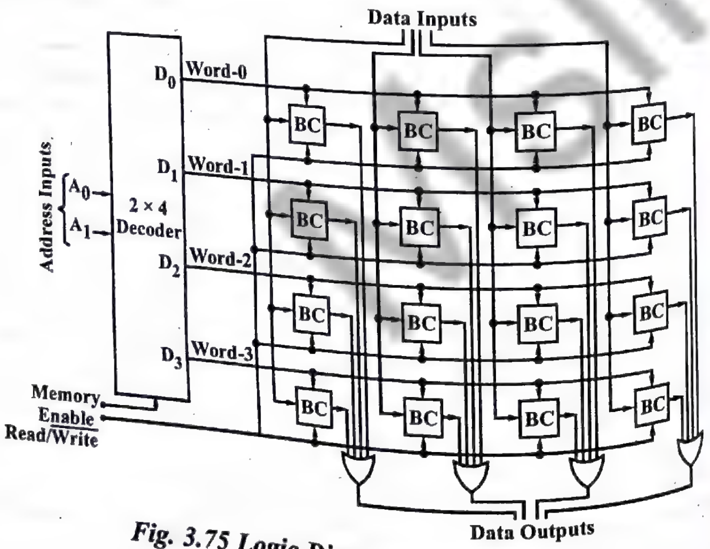


***Fig. 3.75 Logic Diagram of 4 × 4 RAM***

**Q.60. Write short note on flash memory.**

**Or**

**Write short note on flash RAM.** ***(R.G.P.V., Nov. 2018)***

***Ans.*** Flash memory (sometimes called "flash RAM") is a type of constantly-powered nonvolatile memory that can be erased and reprogrammed in units of memory called blocks. It is a variation of Electrically Erasable Programmable Read-Only Memory (EEPROM) which, unlike flash memory, is erased and rewritten at the byte level, which is slower than flash memory updating. Flash memory is often used to hold control code such as the Basic Input/Output System (BIOS) in a personal computer. When BIOS needs to be changed (rewritten), the flash memory can be written to in block (rather than byte) sizes, making it easy to update. On the other hand, flash memory is not useful as Random Access Memory (RAM) because RAM needs to be addressable at the byte (not the block) level.

Flash memory gets its name because the microchip is organized so that a section of memory cells are erased in a single action or "flash". The erasure is caused by Fowler-Nordheim tunneling in which electrons pierce through a thin dielectric material to remove an electronic charge from a floating gate associated with each memory cell. Intel offers a form of flash memory that holds two bits (rather than one) in each memory cell, thus doubling the capacity of memory without a corresponding increase in price.

**Q.61. What is boot strap memory ?** ***(R.G.P.V., June 2014)***

***Ans.*** Many microcomputers and most larger computers do not have their operating system programs stored in ROM. Instead, these programs are stored in external mass memory, usually magnetic disk. How, then, do these computers know what to do when they are powered on ? This small program called a bootstrap program, is stored in ROM. When the computer is powered on, it will execute the instructions that are in this bootstrap program. These instructions typically cause the CPU to initialize the system hardware. The bootstrap program then loads the operating system programs from mass storage into its main internal memory. At that point, computer starts executing the operating system program and is ready to respond to the other user commands. Frequently, this process of start-up is known as ***booting-up*** the system.

**Q.62. What are the different addressing schemes used in memories ? Discuss in specific context – linear versus matrix addressing based on row-column organization.** ***(R.G.P.V., Dec. 2011)***

***Ans.*** To facilitate selection, the memory cells are organized as rectangular array of m-rows and n-columns. When rows and columns are made equal i.e. m = n, then the array becomes a ***Square Array*** of capacity n × n = $n^2$. This type of arrangement is called ***MATRIX Addressing.*** This addressing have the advantage of requiring fewer number of address lines than the number of address lines required by any other rectangular arrangement. This fact is explained below by using an array of 16-cells. Observe carefully the different array arrangements shown in fig. 3.76.

Fig. 3.76 (a) and (b) are equivalent array arrangement as one refers to 8-rows and 2-columns and other refers to 2-rows and 8-columns. Both require 10 address lines (8-rows + 2-columns) to select any of the memory cells. Similarly fig. 3.76 (c) and (d) are equivalent array arrangement and require 17 address lines. Infact 16 lines will suffice, as there is only one column/row. Fig. 3.76 (e) is square array arrangement with equal number of rows & columns and require 8 address lines. So in order to reduce the number of address lines square array arrangement is best.

The arrangement in fig. 3.76 (e) is referred as ***Matrix addressing*** as defined earlier. In contrast the 16 × 1 array arrangement as in fig. 3.76 (c) is called ***Linear Addressing.*** It is because there is only one column and to select any of the cells, one needs to specify the row only.

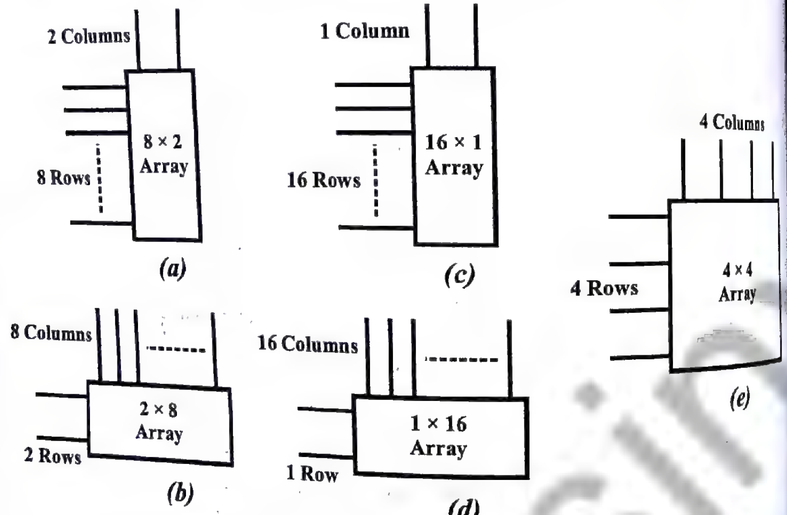


***Fig. 3.76 Different Array Arrangements for 16 Memory Cells***

***Q.63. Write a short note on PLA.*** ***(R.G.P.V., Dec. 2005, June 2007, Dec. 2010, June 2012, May 2018)***

***Or***

***Explain PLA's.*** ***(R.G.P.V., Dec. 2013)***

***Ans.*** A PLA is a simple programmable logic device that consists of a programmable AND array and a programmable OR array. PLA is a type of fixed architecture logic devices with programmable AND gates followed by programmable OR gates. The PLA is used to implement a complex combinational circuit. The AND and OR gates inside the PLA are initially fabricated with fuses among them. The specific Boolean functions are implemented in sum of products (SOP) form by blowing appropriate fuses and leaving the desired connections.

When a manufacture makes a provision to allow programming by the user, the PLA is referred to as a field programmable logic array (FPLA). The AND gates provide the product terms and the OR gates logically sum these product terms and thereby generate a sum of product (SOP) expression. It has n inputs, s product terms and m outputs with s < $2^n$ and can be used to implement a logic function of n variables with m outputs. Since all of the possible $2^n$ minterms are not available, therefore logic minimization is needed to accommodate a given logic function.

The PLA is similar to a ROM in concept except that it does not provide full decoding of the variables and does not generate all the minterms as in the ROM. Thus, in a PLA the decoder is replaced by a group of AND gates, each of which can be programmed to produce a product (AND) term of the input variables. The block diagram of PLA is shown in fig. 3.77.

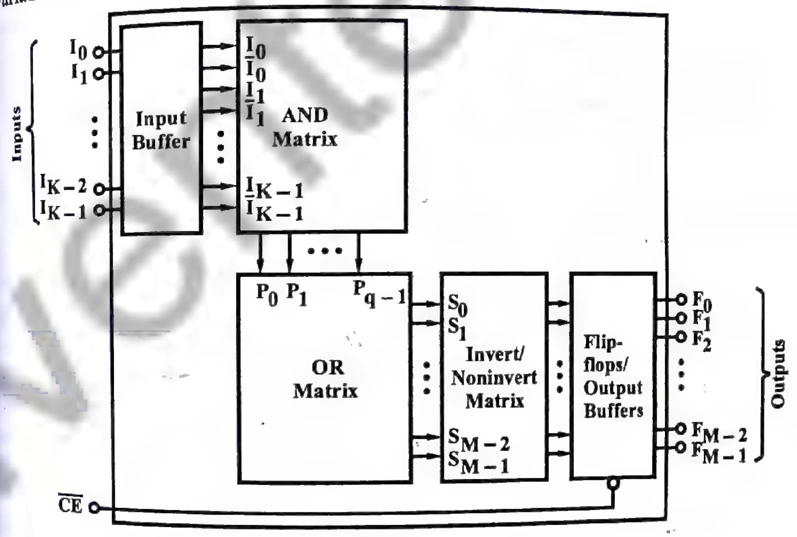
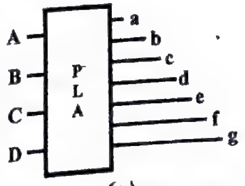


***Fig. 3.77 Block Diagram of PLA***

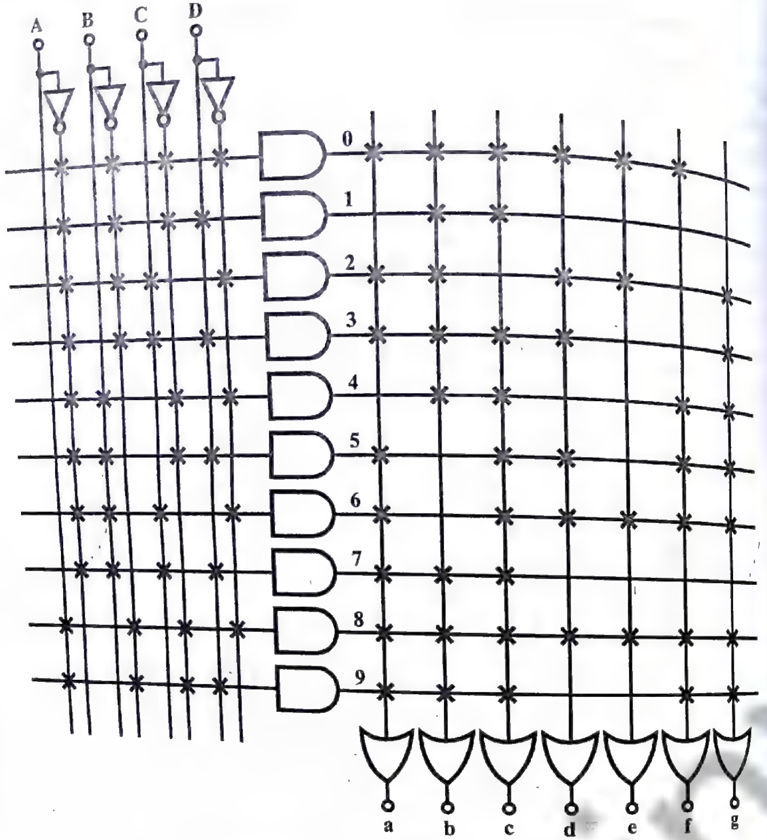
A main advantage of the PLA results from the fact that a wide variety of sequential logic functions are economically obtained by designing for a single modification of the gate mask during circuit fabrication. In the PLA system the feedback loop must be clocked carefully, otherwise some undesirable transient phenomena can develop if we feed the data back directly from the summing matrix into the product matrix.

For example, consider that it is desired to use a PLA to recognize each of the ten decimal digits represented in binary form and to correctly drive a seven-segment display. The PLA must have four inputs, as illustrated in fig. 3.78 (a). Four bits (ABCD) are required to represent the 10 decimal numbers. There must be seven outputs (abcdefg), one output to drive each of the seven-segments of the indicator.



*(a)*

The circuit in fig. 3.78 (b) illustrates the remaining links after programming. The input AND gate array is programmed (fusible links are removed) such that each AND gate decodes one of the decimal numbers.



(b)

*Fig. 3.78 Programmable Logic Array (PLA)*

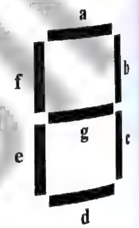


*Fig. 3.79 Seven-segment Display*

Then with the use of fig. 3.79, links are removed from the output OR gate array such that the proper segments of the indicator are illuminated. When ABCD = 0101, segments afgcd are illuminated to display the decimal number 5.

**Application of PLA** – They are used for implementing combinational logic functions and this results in compact circuitry and high switching speed.

***Q.64. What is RAM ? Distinguish between SRAM and DRAM. What is PLA ?*** ***(R.G.P.V., Dec. 2015)***

***Ans.*** Refer the ans. of Q.55, Q.58 and Q.63.

***Q.65. Design a combinational circuit using a ROM. The circuit accepts a 3-bit number and generates an output binary number equal to the square of the input number.*** ***(R.G.P.V., June 2014)***

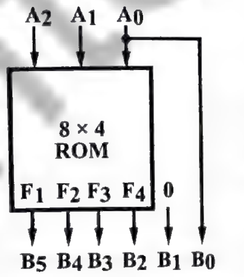
***Or***

***Derive a PLA program table for a combinational circuit that squares a 3-bit number.*** ***(R.G.P.V., June 2017)***

**Table 3.27 Truth Table**

| *Inputs* | | | *Outputs* | | | | | | *Decimal* |
|---|---|---|---|---|---|---|---|---|---|
| $A_2$ | $A_1$ | $A_0$ | $B_5$ | $B_4$ | $B_3$ | $B_2$ | $B_1$ | $B_0$ | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 4 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 9 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 16 |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 25 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 36 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 49 |

***Ans.*** First, we implement the truth table for this combinational circuit as shown in table 3.27. We require 3 inputs and 6 outputs to represent all possible numbers. Note that output $B_0$ is always equal to input $A_0$. Therefore, there is no need to generate $B_0$ with a ROM. Since it is equal to an input variable. Further, as the output $B_1$ is always 0, this output is always known. Hence only four outputs ($B_2B_3B_4B_5$) are to be generated. Therefore, the minimum-size ROM needed must have three inputs and four outputs. As three inputs specify eight combination of 4-bit each, the ROM size must be 8 × 4. The implementation of ROM is shown in fig. 3.80. The other two outputs of the combinational circuit are equal to 0 and $A_0$. The truth table shown in table 3.27 gives all the information needed for programming the ROM, and the block diagram shows the required established connections.



***(a) Block Diagram***

| $A_2$ | $A_1$ | $A_0$ | $F_1$ | $F_2$ | $F_3$ | $F_4$ |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |

***(b) ROM Truth Table***

***Fig. 3.80 ROM Implementation***

## NUMERICAL PROBLEMS

***Prob.12. The capacity of 2K × 16 PROM is to be expanded to 16 K × 16. Find the number of PROM chips required and the number of address lines in the expanded memory.*** ***(R.G.P.V., June 2014)***

***Sol.*** Capacity required = 16 K × 16

Given Chip = 2 K × 16

$$\text{The number of chips} = \frac{16\text{K} \times 16}{2\text{K} \times 16} = 8$$

Total word capacity in the chip = $2 \times 2^{10}$

Hence the address line required for single chip = 11

The word capacity in the expanded memory 16 K = $2^{14}$

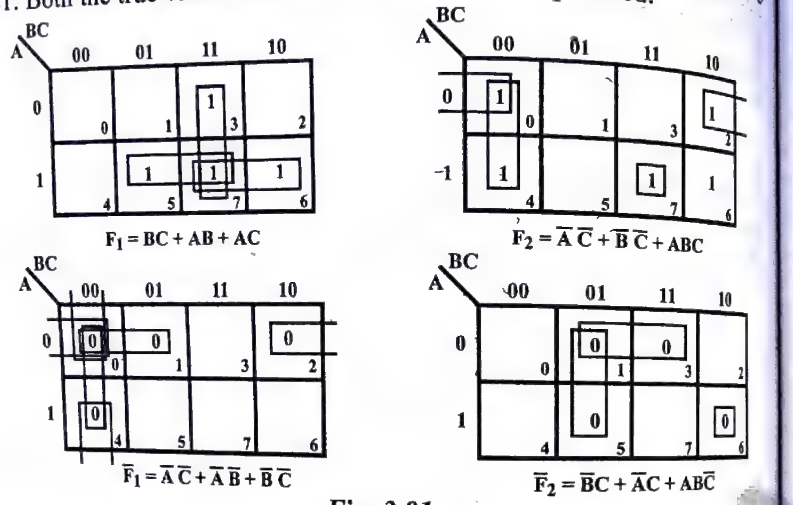
There is the address lines required are 14. **Ans.**

***Prob.13.** A combinational logic circuit is defined by the functions –*

*$F_1 = \Sigma(3, 5, 6, 7)$ and $F_2 = \Sigma(0, 2, 4, 7)$.*

*Implement the circuit with a PLA having three inputs, four product terms and two outputs.* *(R.G.P.V., Dec. 2016)*

***Sol.*** The given functions are simplified in the k-maps as shown in fig. 3.81. Both the true values and the complements are specified.



$F_1 = BC + AB + AC$

$F_2 = \bar{A}\,\bar{C} + \bar{B}\,\bar{C} + ABC$

$\bar{F}_1 = \bar{A}\,\bar{C} + \bar{A}\,\bar{B} + \bar{B}\,\bar{C}$

$\bar{F}_2 = \bar{B}C + \bar{A}C + AB\bar{C}$

*Fig. 3.81*

The combinations which provides a minimum number of product terms are as follows –

$$F_1 = \overline{(\bar{A}\,\bar{C} + \bar{A}\,\bar{B} + \bar{B}\,\bar{C})}$$

$$F_2 = \bar{A}\,\bar{C} + \bar{B}\,\bar{C} + ABC$$

This provides only four distinct product terms i.e., .$\bar{A}\,\bar{C}$, $\bar{A}\,\bar{B}$, $\bar{B}\,\bar{C}$ and ABC. The PLA program table for this combination is shown in table 3.28.

**Table 3.28 PLA Program Table**

| | *Product Term* | *Inputs* | | | *Outputs* | | |
|---|---|---|---|---|---|---|---|
| | | *A* | *B* | *C* | *$F_1$* | *$F_2$* | |
| $\bar{A}\,\bar{C}$ | 1 | 0 | – | 0 | 1 | 1 | |
| $\bar{A}\,\bar{B}$ | 2 | 0 | 0 | – | 1 | – | |
| $\bar{B}\,\bar{C}$ | 3 | – | 0 | 0 | 1 | 1 | |
| ABC | 4 | 1 | 1 | 1 | – | 1 | |
| | | | | | C | T | T/C |

It should be remembered that output $F_1$ is the normal output even though a C is marked under it. This is because $\bar{F}_1$ is produced prior to the output inverter. The inverter complements the function to generate $F_1$ in the output.

The combinational circuit for this problem is very small for practical implementation with a PLA.

# UNIT 4

## INTRODUCTION TO A/D & D/A CONVERTERS & THEIR TYPES, SAMPLE AND HOLD CIRCUITS, VOLTAGE TO FREQUENCY & FREQUENCY TO VOLTAGE CONVERSION

***Q.1.** Draw the circuit diagram of analog to digital converter and explain its working.* *(R.G.P.V., June 2010)*

***Or***

*With the help of circuit diagram explain the A to D converter.* *(R.G.P.V., June 2009, 2012)*

***Or***

*Explain A/D converter and its working.* *(R.G.P.V., Dec. 2013)*

***Or***

*Explain analog to digital converter.* *(R.G.P.V., Dec. 2016)*

***Ans.* A/D Converter** – An analog to digital (A/D) converter takes an analog input voltage and after a certain amount of time produces a digital output code, which represents the analog input. Various types of A/D converters utilize a D/A converter as part of their circuitry. The basic block diagram of A/D converter is illustrated in fig. 4.1.

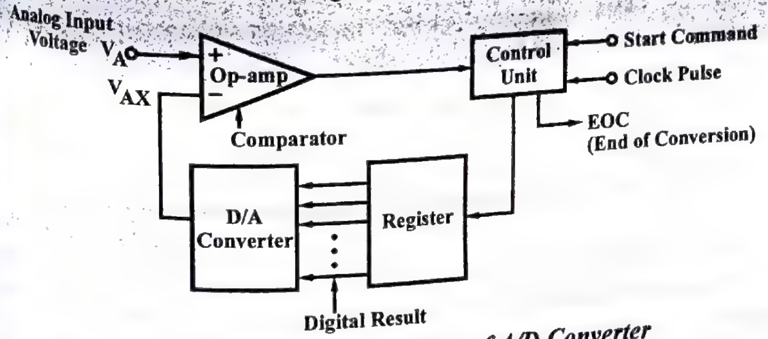


***Fig. 4.1 Basic Block Diagram of A/D Converter***

**Operation** – The start command pulse initiates the operation. The control unit continually modifies the binary number, which is stored in the register at a rate determined by the clock. The binary number in the register is converted to an analog voltage ($V_{AX}$) by the digital to analog (D/A) converter. Comparator compares an analog voltage $V_{AX}$ with the analog input ($V_A$). As long as $V_{AX} <$ $V_A$, the comparator output stays high. When an analog voltage $V_{AX}$ exceeds $V_A$ by at least an amount equal to threshold voltage ($V_{TH}$), the comparator output goes low and stops the process of modifying the register number. An analog voltage $V_{AX}$ is a close approximator to analog input $V_A$ at this point. Within the resolution and accuracy of the system the digital number in the register, which is the digital equivalent of $V_{AX}$ is also the approximate digital equivalent of $V_A$. When the conversion is complete, the control logic activates the end of conversion signal.

The timing for the operation is provided by the input clock signal. The control unit contains the logic circuitry for generating the proper sequence of operation in response to the start command that initiates the process of conversion.

***Q.2. What is the need for A/D converter ?*** ***(R.G.P.V., Dec. 2014)***

***Or***

***What is the need of analog to digital conversion ? (R.G.P.V., Dec. 2015)***

***Ans.*** Data converters convert one form of data into an another form. An A/D converter converts analog data into its equivalent digital data. It is necessary to convert various analog signals like temperature, pressure, flow etc. from various transducers into its equivalent digital data, which in turn act as the input for digital systems.

***Q.3. What are the applications of analog to digital converter ?*** ***(R.G.P.V., June 2015)***

***Ans.*** There are following applications of ADC as given below –

(i) ADC is used in up/down counter.

(ii) ADC is used to convert the analog voltage to a digital form suitable for processing by a digital system.

***Q.4. Write short note on 2-bit simultaneous A/D converter.*** ***(R.G.P.V., Dec. 2010)***

***Or***

***Explain flash A/D converter with circuit diagram and parameters.*** ***(R.G.P.V., June 2011)***

***Ans.*** Simultaneous type analog to digital converter is based on comparing an unknown analog input voltage with a set of reference voltages. To convert an analog signal into a digital signal that has n-bits, $2^n - 1$ comparators are required. For example, the 2-bit A/D converter requires $3(2^2 - 1)$ comparators while a 3-bit converter needs $7(2^3 - 1)$ comparators. The block diagram of 2-bit simultaneous type A/D converter is shown in fig. 4.2.

As shown in fig. 4.2, three op-amps are used as comparators to construct a 2-bit A/D converter. The non-inverting inputs of all the three comparators are connected together to the analog input voltage. The inverting inputs are connected to a set of reference voltages V/4, 2V/4 and 3V/4 obtained using a resistive divider network.

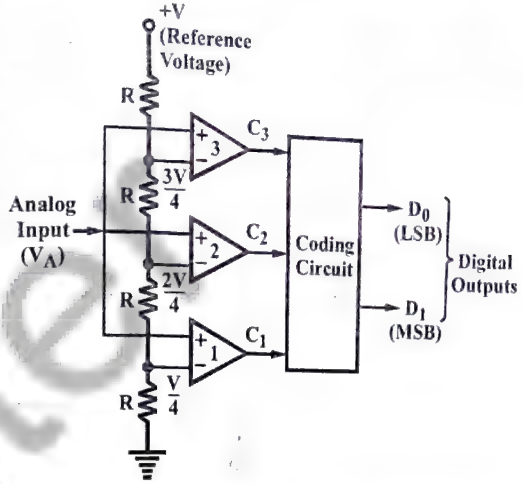


***Fig. 4.2 Block Diagram of 2-bit Simultaneous Type A/D Converter***

It is better to recall that the comparator output is in HIGH state when the voltage at the non-inverting input terminal is higher than the voltage at the inverting terminal and in LOW state otherwise. Now, when the analog input voltage is less than V/4, the voltage at the non-inverting terminals of all the three comparators is less than the respective reference voltage, and thereby the comparator outputs $C_1$ $C_2$ $C_3$ = 000. When analog input is between V/4 and V/2, the comparator output $C_1$ $C_2$ $C_3$ = 100 as shown in table 4.1. It also shows the comparator outputs for other ranges of analog voltage and their corresponding digital outputs.

**Table 4.1 Comparator and Digital Outputs for a 2-bit Simultaneous Type A/D Converter**

| *Analog Input Voltage ($V_A$)* | *Comparator Outputs* | | | *Digital Outputs* | |
|---|---|---|---|---|---|
| | *$C_1$* | *$C_2$* | *$C_3$* | *$D_1$* | *$D_0$* |
| $0 \le V_A \le V/4$ | 0 | 0 | 0 | 0 | 0 |
| $V/4 < V_A \le V/2$ | 1 | 0 | 0 | 0 | 1 |
| $V/2 < V_A \le 3\ V/4$ | 1 | 1 | 0 | 1 | 0 |
| $3V/4 \le V_A \le V$ | 1 | 1 | 1 | 1 | 1 |

Since, there are four ranges of analog input voltages, this can be coded as 2 digital output bits ($D_1$, $D_0$). In order to encode three comparator outputs into two digital outputs, the coding circuit can be designed as follows –

Logic expressions for $D_1$ and $D_0$ can be written as

$$D_1 = C_1C_2\overline{C}_3 + C_1C_2C_3$$

$$= C_1C_2(\overline{C}_3 + C_3) = C_1C_2$$

$$D_0 = C_1\overline{C}_2\overline{C}_3 + C_1C_2C_3 = C_1(\overline{C_2 \oplus C_3})$$

using the simplified expressions of $D_1$ and $D_0$, the logic diagram of a 2-bit simultaneous type A/D converter is drawn as shown in fig. 4.3.

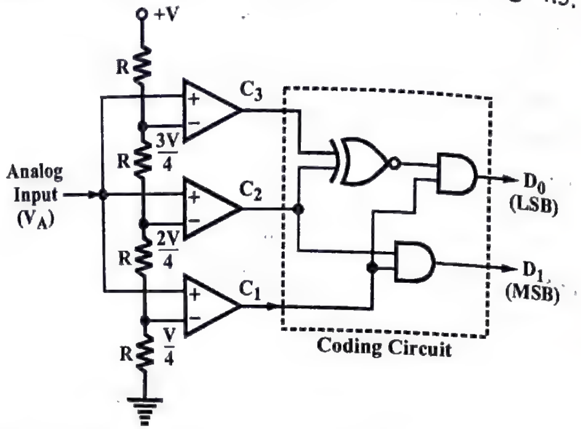


*Fig. 4.3 Logic Diagram of 2-bit Simultaneous Type ADC*

***Q.5. Describe the working of digital ramp A/D converter. Also draw its block diagram.***

*Ans.* The one of the simplest form of basic A/D converter uses a binary counter as the register and allows the clock to increment the counter one step at a time until $V_{AX} \geq V_A$. It is called a digital ramp A/D converter since the waveform at $V_{AX}$ is a step by step ramp. The block diagram for a digital ramp A/D converter is illustrated in fig. 4.4. It contains a counter, a D/A converter, an analog comparator and a control AND gate. The output of comparator serves as the active low end of conversion signal. When we assume that the analog voltage to be converted is positive then the operation performed is as follows –

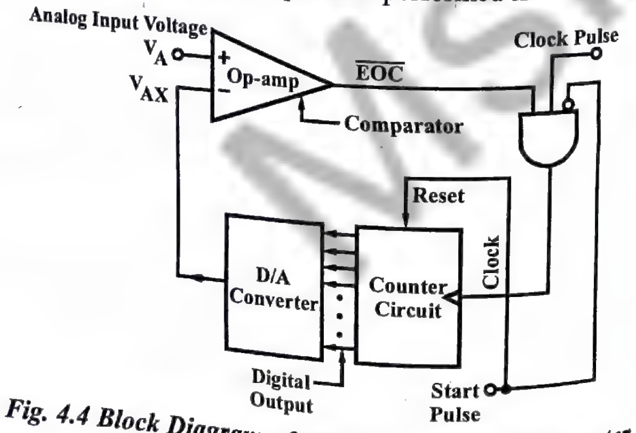


*Fig. 4.4 Block Diagram of Digital Ramp A/D Converter*

**Working** – A start pulse is applied to reset the counter to zero. The high at start also inhibits clock pulses from passing through the AND gate into the counter. With all zeros at its input the D/A converters output will be $V_{AX} = 0V$. Because $V_A > V_{AX}$, the output of comparator $\overline{EOC}$ will be high. When start returns low, the AND gate is enabled and clock pulses get through to the counter. As the counter advances, the output D/A converter ($V_{AX}$) increases one step at a time. This continues until $V_{AX}$ reaches a step that exceeds $V_A$ by an amount equal to or greater than $V_{TH}$. At this point $\overline{EOC}$ will go low and inhibit the flow of pulses into the counter and the counter will stop counting. Now the conversion process is complete as signaled by the high to low transition at $\overline{EOC}$ and the contents of the counter are the digital representation of $V_A$.

The counter will hold the digital value until the next start pulse initiates a new conversion.

***Q.6. Draw the circuit of a binary ladder network A/D converter and explain its working.*** ***[R.G.P.V., Dec. 2003(EL/ET)]***

*Ans.* An A/D converter that uses an up/down counter is shown in fig. 4.5. It is an up/down counter and has the up and down count control lines in addition to the advance line at its input. The output of ladder is fed into a comparator, which has two outputs instead of one as before. When the analog voltage is more positive than the ladder output, the up output of the comparator is high. When the analog voltage is more negative than the ladder output the down counter is high.

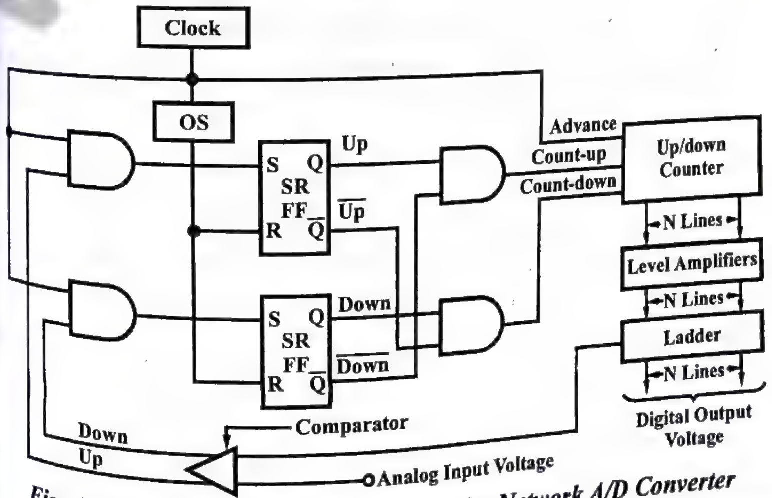


*Fig. 4.5 Block Diagram of Binary Ladder Network A/D Converter*

If the up output of the comparator is high then the AND gate at the input of the up flip-flop is open and the first time the clock goes positive the up flip-flop is set. For the moment if we assume that the down flip-flop is reset, the AND gate which controls the count-up line of the counter will be true and the counter will advance one count.

The counter can advance only one count because the output of the one-shot resets both the up and down flip-flops just after the clock goes low. Then this can be considered as one count-up conversion cycle. We note that the AND gate that controls the count-up line has inputs of up and $\overline{\text{down}}$. In a similar manner the count-down line AND gate has inputs of down and $\overline{\text{up}}$. This could be considered as EX-OR arrangement and ensures that the count-down and count-up lines cannot both be high at the same time. As long as the up line out of the comparator is high, the converter continues to operate one conversion cycle at a time.

At the time where the ladder voltage becomes more positive than the analog input voltage, the up line of the comparator goes low and the down line goes high. Then the converter goes through a count-down conversion cycle. The ladder voltage is within 1 LSB of the analog voltage at this point and the converter oscillates about this point. This is not desirable because we want the converter to ease operation and not jump around the final value to adjust the comparator such that its outputs do not change at the same time.

Here we can accomplish this by adjusting the comparator such that the up output will not go high unless the ladder voltage is more than 1/2 LSB below the analog voltage. In a similar way, the down output will not go high unless the ladder voltage is more than 1/2 LSB above the analog voltage. This is known as entering on LSB and provides a digital output which is within 1/2 LSB.

***Q.7. Describe the successive approximation A/D converter with the help of necessary diagram and waveforms.*** ***(R.G.P.V., June 2008, 2009, Dec. 2010)***

***Or***

***Explain with the help of block diagram any one type of analog to digital converter.*** ***(R.G.P.V., Dec. 2012)***

***Or***

***With a neat diagram explain successive approximation type A/D converter in detail.*** ***(R.G.P.V., Dec. 2014)***

***Or***

***Explain successive approximation techniques for analog to digital conversion.*** ***(R.G.P.V., Feb. 2010, June 2015)***

***Or***

***Explain any one type of analog to digital converter in detail.*** ***(R.G.P.V., May 2018)***

***Ans.*** The successive approximation A to D converter is one of the most widely used type of A/D converter. It has much shorter conversion time than the other types. It also has a fixed conversion time, which is not dependent on the value of the analog input. A basic block diagram of a 4-bit successive approximation A/D converter is shown in fig. 4.6. It consists of a D/A converter, an output register, a comparator and control logic.

**Operation** – The bits of the D/A converter are enabled one at a time, starting with MSB. As each bit is enabled, the comparator produces an output that indicates whether the analog input voltage is greater or less than the output of D/A converter $V_{AX}$. If the output of D/A converter is greater than the analog input, the comparator output is low and resulting the bit in the control register to reset. If the output of D/A converter is less than the analog input, the comparator output is high and the bit is retained in the control register.

The system first enables the MSB, the next significant bit and so on. After all the bits of the D/A converter have been tried, the conversion cycle is complete. The process of each bit takes one clock cycle, so the total conversion time for an N-bit successive approximation type A/D converter will be N clock cycles. It is expressed as –

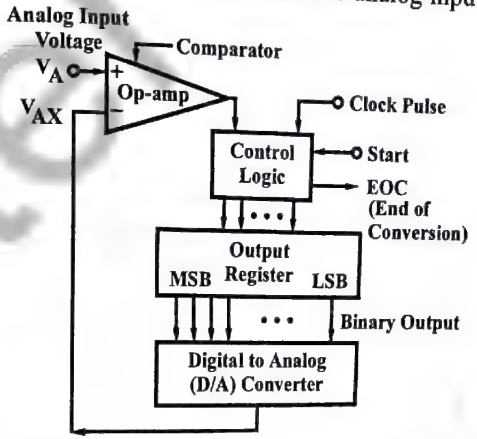


*Fig. 4.6 Basic Block Diagram of Successive Approximation Analog to Digital Converter*

$T_c = (N \times 1)$ clock cycles (for successive approximation converter)

The conversion time will be the same regardless of the value of $V_A$. This is because the control logic has to process each bit to observe whether a 1 is required or not.

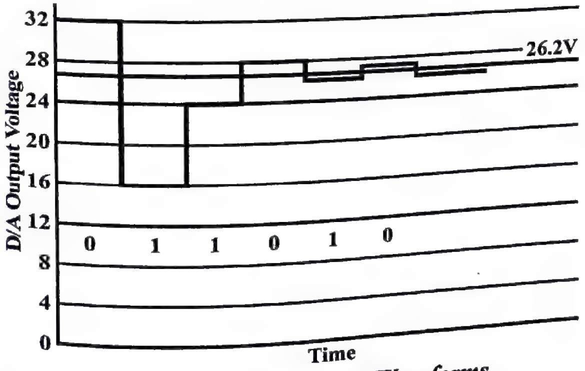


*Fig. 4.7 D/A Converter Waveforms*

Fig. 4.6 shows the block diagram of the system and fig. 4.7 shows the output of the D/A converter compared with the input voltage.

Output of the D/A converter compared with the input voltage. Assume a five-bit conversion must be made for a 26.2V signal using a maximum reference voltage of 64 V. Referring to the flow diagram, the register is first set to 0, then a 1 is placed in its MSB. This is converted by the D/A and fed to the comparator where it is compared with the input voltage. If the D/A output exceeds the input, the bit is set to a 0; if the D/A is less than the input, the 1 is retained in the register. In this case, the output of the D/A is 32 V, resulting in this bit being set to 0.

Next, a 1 is placed in the next bit to the right, resulting in an output of 16 V from the D/A. Since this is less than the input, the 1 is retained in this bit position. Placing a 1 in the third position results in a 24 V output, again less than the input. Therefore, the 1 is retained. Placing a 1 in the fourth position results in an output of 28 V, exceeding the input. Therefore, bit 4 is set to 0. A 1 in bit 5 results in 26 V, less than the input; it is therefore retained. A 1 in bit 6 results in 27 V output, exceeding the input; it is therefore set to 0. Thus, the resultant binary word is 011010, representing 26 V.

The successive approximation method is very fast; many units convert in less than 250 ns/bit. However, it is more expensive than the ramp methods.

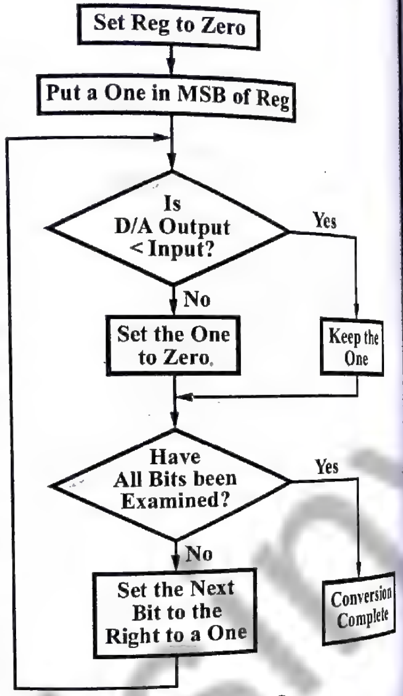


Fig. 4.8

**Q.8. Why analog to digital converters is needed ? Explain any one digital converters.** *(R.G.P.V., Dec. 2017)*

*Ans.* Refer the ans. of Q.2 and Q.7.

**Q.9. Enlist the various types of analog to digital (A/D) converter and explain any one of them with neat sketch.** *(R.G.P.V., Dec. 2015)*

*Ans.* There are various types of A/D converters as listed below –

(i) Simultaneous or flash type A/D converter
(ii) Counter type A/D converter
(iii) Continuous type A/D converter
(iv) Successive approximation type A/D converter
(v) Single slope type
(vi) Dual slope type
(vii) Voltage to frequency (integrating) type

Also refer the ans. of Q.7.

**Q.10. What are the merits and demerit of successive approximation type A/D converter ?**

*Ans.* **Merits** – Main merits are –

(i) The main merit of the successive approximation type A/D converter is speed.

(ii) It takes only n-clock pulses to produce n-bit resolution of the analog signal.

(iii) It has a big improvement over the counter method.

(iv) It can be accomplished through the hardware and software.

(v) The successive approximation method is still considerably better than the counter method even with slower designs.

**Demerit** – It requires a digital to analog converter.

**Q.11. Why is the successive approximation type of ADC faster than the counting type ?**

*Ans.* The number of clock cycles required for conversion of any analog sample is fewer in successive approximation type of ADC than in the case of a counting type ADC so the former is faster than the latter one.

**Q.12. Describe the working of dual slope A to D converter.** *[R.G.P.V., Dec. 2002 (CS)]*

*Ans.* The functional diagram of a dual slope A/D converter is illustrated in fig. 4.9. A dual slope A/D converter uses an operational amplifier to integrate the analog input. The output of the integrator is a ramp, whose slope is proportional to the input signal because the components R and C are fixed. If the ramp is allowed to continue for a fixed time then the voltage it reaches in that time depends on the slope of the ramp and therefore on the value of input signal.

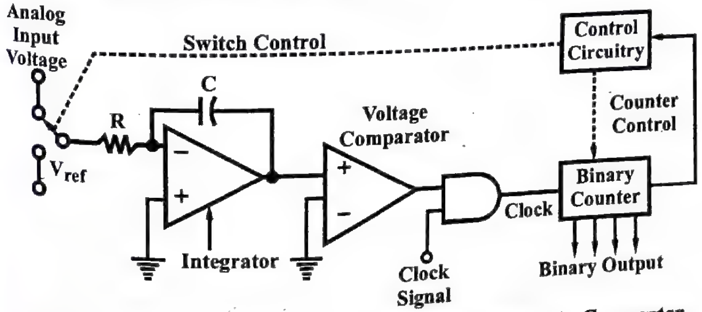


Fig. 4.9 Functional Diagram of Dual Slope A/D Converter

**Operation** – The conversion starts with the switch connected to the analog input. We assume that the input is a negative voltage and it is constant for a period of time, so, the output of the integrator is a positive ramp. The ramp is allowed to continue for a fixed time and the voltage it reaches in that time is directly dependent on the analog input. When the counter reaches a particular count the fixed time is controlled by sensing the time. At that time the counter is reset the control circuitry causes the switch to be connected to a reference voltage with having a polarity opposite to that of the analog input and in this case a positive reference voltage. Therefore, the output of the integrator is a negative going ramp starting from the positive value it reached during the first integration.

The AND gate is enabled and the counter starts counting. When the ramp reaches zero volts, the voltage comparator switches to low inhibiting the clock pulses and the counter stops counting. The binary count is latched thus completing one conversion. The count it contains at that time is proportional to the time required for the negative ramp to reach zero, which is proportional to the positive voltage reached during the first integration, which in turn proportional to the analog input. The accuracy of converter does not depend on the values of the integrator components or upon any changes in them. The reference voltage $V_{ref}$ should be very precise because the accuracy does depend on $V_{ref}$.

***Q.13. What are merits and demerits of a dual slope A/D converter?***

***Ans.*** **Merits** – The various merits of a dual slope A/D converter are given as under –

(i) The dual slope A/D converter is one of the slowest converter.

(ii) It does not need precision components such as a D/A converter or voltage controlled oscillator (VCO).

(iii) It has low sensitivity to noise and to variation in its component values caused by temperature change.

(iv) The accuracy of converter does not depend on the values of the integrator components.

**Demerits** –

(i) It requires large conversion time.

(ii) It cannot be used in any data acquisition applications.

(iii) It is restricted to use signals having low to medium frequencies.

***Q.14. Distinguish single slope and double slope A/D converter.***
***(R.G.P.V., Dec. 2014)***

***Ans.*** The main unit of single slope converter is a ramp generator whose O/P voltage begins at 0 with 0V and increases linearly with time up to a maximum voltage $V_m$ on receiving a RESET from a control circuit.

In dual slope type A/D converter, the integrator generates two different ramps, viz one with unknown analog input voltage as the input and another with a known reference voltage as the input. Hence, it is called dual slope type A/D converter.

***Q.15. Define the following specification of A/D converter –***
***(i) Input voltage range***
***(ii) Conversion time***
***(iii) Digital output formate.***

***Ans. (i) Input Voltage Range*** – The range of input voltage for an ADC signifies the input voltage which can be converted by the ADC to digital code.

***(ii) Conversion Time*** – The time required for conversion of analog sample to corresponding digital code is known as the conversion time of an ADC.

***(iii) Digital Output Formate*** – Digital output format of an ADC means a format, such as straight binary, 1's complement, 2's complement, complementary straight binary etc; in which the digital output is available.

***Q.16. State maximum conversion time and average conversion time.***
***(R.G.P.V., Dec. 2013)***

***Ans.*** The conversion time $t_c$ is the time gap between the end of the START pulse and the activation of the $\overline{EOC}$ output. The counter begins counting from 0 and counts up until $V_{AX}$ is more than $V_A$, at which point $\overline{EOC}$ goes LOW to finish the conversion process. Clearly, the value of conversion time, $t_c$ relies on $V_A$. A greater value will require more steps before the staircase voltage is more than $V_A$.

The maximum conversion will take place when $V_A$ is just below full scale so that $V_{AX}$ must go to the last step to activate $\overline{EOC}$. For an N-bit converter this will be

$$t_c(\max) = (2^N - 1) \text{ clock cycles.}$$

The average conversion can be defined as the half of the maximum conversion time. For the digital ramp converter, this would be,

$$t_c(\text{avg}) = \frac{t_c(\max)}{2}$$

$$\approx 2^{N-1} \text{ clock cycles}$$

***Q.17. Write short note on D/A converter.***
***(R.G.P.V., Dec. 2002 (S), 2004, May/June 2006, 2009)***

***Ans.*** The process of conversion of a digital signal to analog signal is referred to as a digital to analog conversion. The system used for realizing this conversion is referred to as a digital-to-analog converter. An integral part of the D/A converter there must be a register that can be used to store the digital information. The simplest register is formed by use of S-R flip-flops, with one flip-flop per bit. There must also be level amplifiers between the registers and the resistive network to ensure that the digital signals presented to the network are all of the same level and are constant. There must be some form of gating on the input of the register such that the flip-flops can be set with the proper information from the digital system.

The block diagram of 4-bit D/A converter is illustrated in fig. 4.10. The level amplifiers have two inputs, such as one input is the + 10V from the precision voltage source and the other is from a flip-flop. The amplifiers work in such a way that when the input from a flip-flop is high, the output of the amplifier is at + 10V. When the input from the flip-flop is low, the output is zero volt.

The four flip-flops form the register necessary for storing the digital information. The flip-flop on the right shows the MSB and the flip-flop on the left shows the LSB. Each flip-flop is a simple S-R latch and needs a positive level at the S or R input to set or reset. When the strobe pulse line goes high (or 1), only one of the two gate outputs connected to each flip-flop is high and the flip-flop is set or reset accordingly. Thus the data are entered into the register each time the strobe pulse occurs.

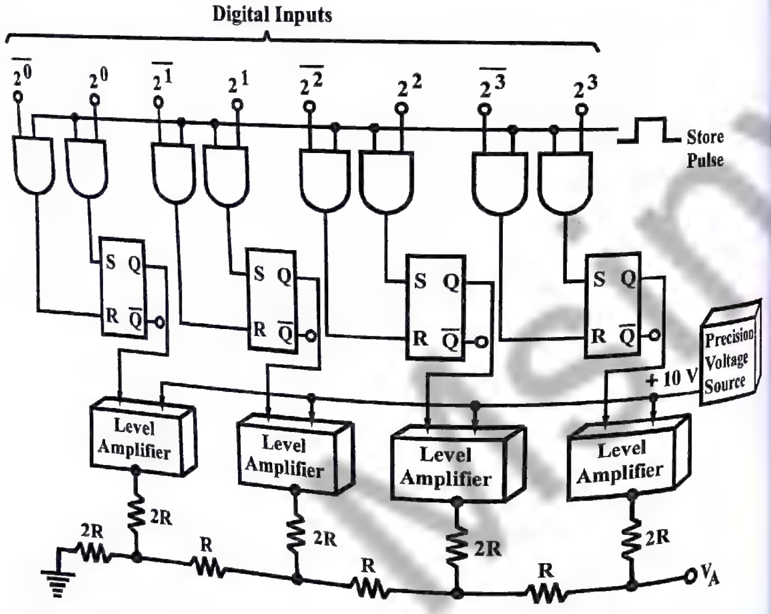


*Fig. 4.10 Block Diagram of Four-bit Digital to Analog Converter*

***Q.18. Explain quantization error.*** ***(R.G.P.V., June 2014)***

***Ans.*** In A/D conversion, a continuous analog voltage is represented by an equivalent set of digital numbers. When the digital numbers are converted back to analog voltage by a DAC, the output is a staircase waveform, which is a discontinuous signal, composed of a number of discrete steps. The smallest digital step is due to the LSB and it can be made smaller only by increasing the number of bits in the detail representation. This error is called quantization error or digitizing error.

***Q.19. What is a bipolar D/A converter.*** ***(R.G.P.V., June 2014)***

***Ans.*** Up to this point we have assumed that the binary input to a DAC has been an unsigned number and that the DAC output has been a positive voltage or current. Many DACs are designed to produce both positive and negative values, such as –10 to +10 V. This is generally done by using the binary input as a signed number with the MSB as the sign bit. Negative input values are often represented in 2's complement form, although the true magnitude form is also used by some DACs. For example, suppose that we have a six bit bipolar DAC that uses the 2's complement system and has a resolution of 0.2 V. The binary input values range from 100000 (–32) to 011111 (+31) to produce analog outputs in the range from – 6.4 to +6.2 V.

***Q.20. Draw and explain the binary weighted resistor D/A converter.***

***Ans.*** The binary weighted resistor D/A converter is shown in fig. 4.11. The operational amplifier summing circuit is used to produce a weighted sum of the digital inputs, where the weights are proportional to the weights of the bit position of inputs. The $V_{ref}$ is an accurate reference voltage and the resistors are precision resistors to get accurate input currents. When all switches are open, all input and output currents are zero. When all switches are closed, the values of input currents are obtained as –

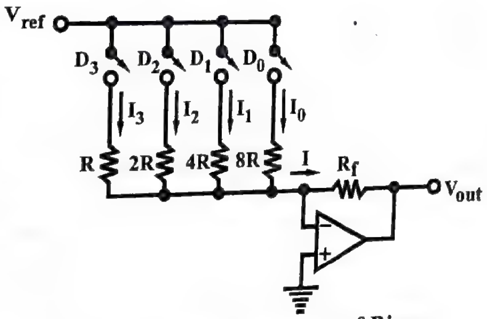


*Fig. 4.11 Circuit Diagram of Binary Weighted Resistor D/A Converter*

$$I_3 = \frac{V_{ref}}{R},\ I_2 = \frac{V_{ref}}{2R}$$

$$I_1 = \frac{V_{ref}}{4R} \text{ and } I_0 = \frac{V_{ref}}{8R}$$

The value of output current with all switches closed is the sum of all input c urrents.

$$I_{out} = \frac{V_{ref}}{R}\left(1+\frac{1}{2}+\frac{1}{4}+\frac{1}{8}\right) = 1.875\left[\frac{V_{ref}}{R}\right]$$

If 0 (zero) is used for an open switch and 1(one) for a closed switch, then we can rewrite as –

$$I_{out} = \frac{V_{ref}}{R}(D_3 + 0.5D_2 + 0.25D_1 + 0.125D_0)$$

It can be equivalently written in the power of 2 as –

$$I_{out} = \frac{V_{ref}}{R}(D_3 + 2^{-1}D_2 + 2^{-2}D_1 + 2^{-3}D_{0)}$$

Here we observe that the output current is the sum of binary weighted input currents. The output voltage ($V_{out}$) is given by the relation as –

$$V_{out} = -\left(D_3 + \frac{D_2}{2} + \frac{D_1}{4} + \frac{D_0}{8}\right) \times \left(\frac{R_f}{R}\right)$$

***Q.21. What are the demerits of a binary weighted resistor D/A converter?***

***Ans.*** Demerits of binary weighted D/A converter are given as follows-

(i) The different valued precision resistor must be used for each bit position of the digital input.

(ii) For a weighted resistor circuit to be monotonic the tolerance of the resistors must be less than the percent resolution.

(iii) As the number of bits increases, the range of resistance values gets awkward.

(iv) Because of the tolerance and range problems the mass production of weighted resistor D/A converters is impractical.

(v) If resolution is about four percent, the resistors require a tolerance of better than ±0.4 % for a monotonic output.

(vi) The MSB resistor is needed to handle a much greater current through the LSB resistor.

***Q.22. Describe the practical D/A converter.*** ***(R.G.P.V., Dec. 2008)***

***Or***

***Explain a 4 bit R-2R ladder type D/A converter in detail.*** ***(R.G.P.V., Dec. 2014)***

***Or***

***Explain the operation of R-2R ladder type digital to analog (D/A) converter with a neat sketch.*** ***(R.G.P.V., Dec. 2015)***

***Ans.*** The R-2R ladder type D/A converter is most popular. This circuit uses a ladder network containing series parallel combinations of two resistors of the values R and 2R. To prevent loading the operational amplifier is used in the voltage follower configuration. The circuit diagram of a R-2R ladder type D/A converter with a 4-bit input is shown in fig. 4.12. When a digital signal $D_3$ $D_2$ $D_1$ $D_0$ is applied at the input terminals of the D/A converter, an equivalent analog signal is produced at the output terminal.

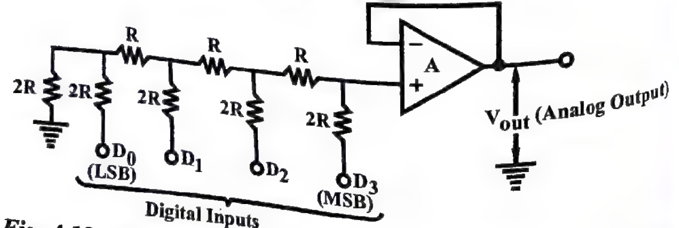
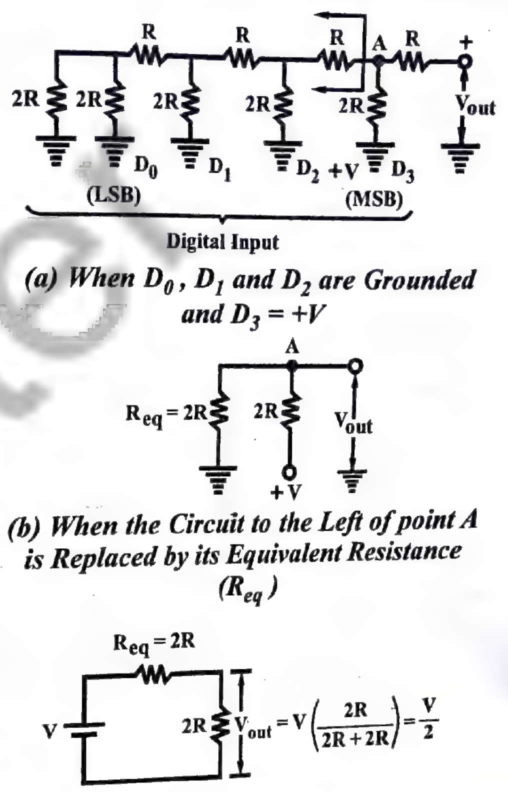


***Fig. 4.12 Circuit Diagram of R-2R Ladder Type D/A Converter***

**Operation** – The operation of D/A converter for the input 1000 is illustrated in fig. 4.13. At the left end of the ladder the resistor 2R is in parallel with resistor 2R, so the combination of both is equivalent to resistor R. This resistor R is in series with another resistor R giving resistor 2R. This 2R resistor in parallel with another 2R resistor, so the combination of both is equivalent to resistor R. This process in continues and we obtain the equivalent resistance $R_{eq}$ = 2R. The value of output voltage, $V_{out}$ = V/2.



***(a) When $D_0$, $D_1$ and $D_2$ are Grounded and $D_3$ = +V***

***(b) When the Circuit to the Left of point A is Replaced by its Equivalent Resistance ($R_{eq}$)***

***(c) Determination of Output Voltage ($V_{out}$) using the Voltage Divider Method***

***Fig. 4.13 Representation of Output Voltage ($V_{out}$) When the Input is 1000***

***Q.23. What are the advantages of R-2R ladder type D/A converter.***

***Ans.*** The advantages of R-2R ladder type D/A converter over binary weighted resistor type D/A converter are as follows –

(i) The main advantage of R-2R ladder type D/A converter is that the resistors of only two values are needed. So that the standard resistors can be used.

(ii) In high resolution D/A converter the problem of the large difference in resistor values between the LSB and the MSB does not occur in R-2R ladder type D/A converter.

(iii) The R-2R ladder type D/A converter is to use a circuit which uses resistances that are fairly close in value.

**Q.24. Describe the operation of switched current source type D/A converter.**

***Or***

**With the help of block diagram explain any one type of digital to analog converter.** **(R.G.P.V., Dec. 2012)**

*Ans.* Fig. 4.14 illustrates the 4-bit switched current-source type D/A converter. Here we note that an R-2R ladder is connected to a voltage source ($V_{ref}$). The current is the first 2R resistor from supply is given by $I_3 = V_{ref}/2R$, because voltage source $V_{ref}$ is directly connected across 2R resistor. The current in the second 2R resistor is given as $I_2 = V_{ref}/4R$, because the equivalent resistance to the right of the second 2R resistor is 2R and so the current $V_{ref}/(R+R)$ coming into the first R resistor is equally divided between the second 2R resistor and the 2R resistor to its right.

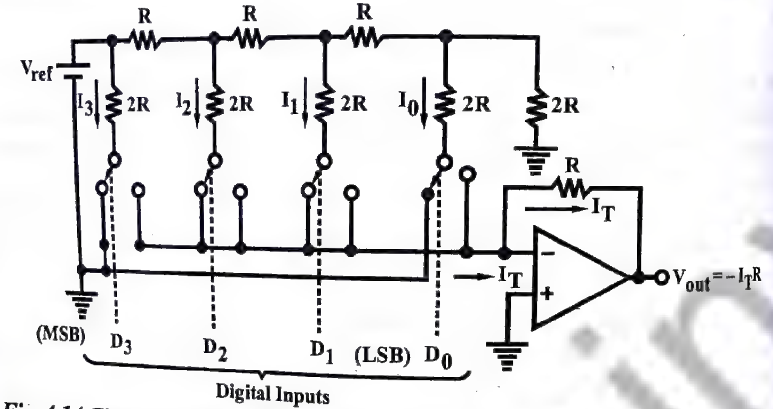


*Fig. 4.14 Circuit Diagram of Four-bit Switched Current Source D/A Converter*

Generally, the current which flows in each 2R resistor is given by the relation as –

$$I_n = \left(\frac{V_{ref}}{R}\right) \times \frac{1}{2^{M-n}}$$

where $n = 0, 1, 2, ....., M - 1$

$M =$ Number of inputs.

The switches that connect the currents either to ground or to the input of the op-amp are controlled by the digital input. The op-amp sums up all those currents whose corresponding digital inputs are high (or 1). The op-amp also serves as voltage to current converter. It is connected in an inverting configuration and its output is $V_{out} = -I_TR$, here $I_T$ is the sum of the currents that have been switched to its input. If $V_{ref}$ is an externally variable voltage, the output of the D/A converter is proportional to the product of the variable $V_{ref}$ and the variable signal input. In that case, the circuit is known as a multiplying D/A converter and the output represents the product of analog input $V_{ref}$ and a digital input.

**Q.25. What are applications of D/A converter ?**

*Ans.* The D/A converters are used whenever the output of a digital circuit must provide an analog voltage or current to drive an analog device. The some of the most common applications are explained as follows –

***(i) Control*** – The digital output from a computer can be converted to an analog control signal to adjust the speed of motor, the temperature of a furnace and to control almost any physical variable.

***(ii) Automatic Testing*** – The computers can be programmed to generate the analog signals through a D/A converter required to test analog circuitry. Normally, the test circuits analog output response will be converted to a digital value by an A/D converter and connect into the computer to be stored, displayed and analyzed.

***(iii) Reconstruction of Signal*** – An analog signal is digitized in many applications, i.e., the successive points on the signal are converted to their digital equivalents and stored in memory. This is performed by analog-to-digital converters. A digital-to-analog converter can then be used to convert the stored digitized data back to analog one point at a time and thereby reconstructing the original signal. This combination of digitizing and reconstructing is used in digital storage oscilloscopes (DSO), audio compact disk systems and digital audio and video recording.

***(iv) Serial D/A Converters*** – The serial D/A converters are now readily available with a built-in serial-in-parallel-out shift register. Many of these devices have more than one D/A converter on the same chip. The digital data along with a code which specifies that D/A converter are sent to the chip one bit at a time. As each bit is presented on the D/A converter input and a pulse is applied to the serial clock input to shift the bit in and then after the proper number of clock pulses the data value is latched and then converted to its analog value.

**Q.26. Discuss the performance characteristics of D-to-A converter.** **(R.G.P.V., Nov./Dec. 2007, Feb. 2010)**

***Or***

**What are the performance characteristics of D/A converter ?** **(R.G.P.V., June 2014)**

*Ans.* The performance characteristics of D-to-A converter is given by following parameters –

***(i) Resolution (Step Size)*** – The resolution of a DAC is defined as the smallest change that can occur in an analog output as a result of a change in the digital input. The resolution of a DAC is also defined as the reciprocal of the number of discrete steps in the full-scale output of the DAC. The resolution or step size is the size of the jumps in the staircase waveform.

$$\% \text{ resolution} = \frac{\text{Step size}}{\text{Full scale}} \times 100\%$$

Since, $\quad$ Full scale = Number of steps × Step size

Therefore, $\quad \% \text{ resolution} = \dfrac{1}{\text{Total number of steps}} \times 100\%$

For N-bit DAC, the number of different levels will be $2^N$ and the number of steps will be $2^N - 1$. The greater the number of bits, the greater will be the number of steps and the smaller will be the step size, and therefore, the finer will be the resolution. And also, the cost of the DAC will increase with the number of input bits.

***(ii) Accuracy*** – The accuracy of a DAC is usually specified in terms of its full-scale error and linearity error, which are normally expressed as a percentage of the converter's full-scale output. Full scale error is the maximum deviation of the DAC's output from its expected (ideal) value, expressed as a percentage of the full scale. Linearity error is the maximum deviation of the analog output from the ideal output. The accuracy and resolution of a DAC must be compatible.

***(iii) Settling Time*** – The operating speed of a DAC is usually specified by giving its settling time. It is defined as the total time between the instant when the digital input changes and the time that the output enters a specified error band for the last time, usually ±1/2 LSB around the final value after the change in digital input. Generally, DACs with a current output will have shorter settling times than those with voltage outputs.

***(iv) Offset Voltage*** – Ideally, the output of a DAC should be zero when the binary input is zero. But there is a very small output voltage under this situation called the ***offset voltage***. This offset error, if not corrected, will be added to the expected DAC output for all input cases.

***(v) Monotonicity*** – A DAC is said to be monotonic if its output increases as the binary input is incremented from one value to the next. This means that the staircase output will have no downward steps as the binary input is incremented from 0 to full-scale value. The DAC is said to be non-monotonic if its output decreases when the binary input is incremented.

***Q.27. How can we describe the resolution of a digital to analog converter?***
***(R.G.P.V., June 2015)***

***Ans.*** Refer the ans. of Q.26 (i).

***Q.28. Explain the transfer characteristics and various performance parameters of DAC.*** ***(R.G.P.V., June 2015)***

***Ans.*** The transfer characteristics of 4-bit DAC is shown in fig. 4.15. In the transfer characteristics, analog output voltage $V_o$ is plotted against all 16 possible inputs.

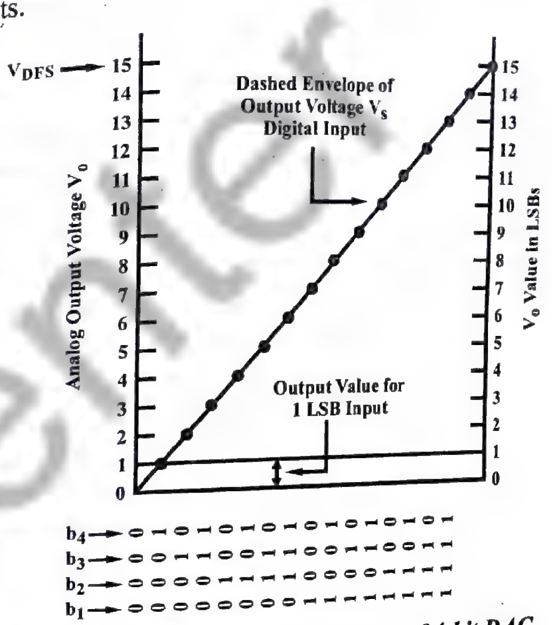


***Fig. 4.15 Transfer Characteristics of 4-bit DAC***

**Performance Parameters of DAC** – Refer the ans. of Q.26.

***Q.29. Discuss about the sample and hold circuits.***
***(R.G.P.V., June 2003, Dec. 2008, June 2009, Nov. 2018)***

***Or***

***Write short note on sample and hold circuits.***
***(R.G.P.V., June 2006, 2007, 2008, 2010, Dec. 2010, 2016, June 2017)***

***Or***

***Discuss about sample and hold circuits in A/D converter.***
***(R.G.P.V., June 2011)***

***Or***

***Explain the principle working of sample and hold circuits.***
***(R.G.P.V., Dec. 2012)***

***Or***

***Draw the circuit diagram of sample and hold circuit and explain its working.*** ***(R.G.P.V., Dec. 2015)***

*Ans.* The sample and hold circuit samples an input signal and holds on to its last sampled va'ue until the input is again sampled. The sample and hold circuit using an operational amplifier with an E-MOSFET is as shown in fig. 4.16. The E-MOSFET in this circuit works as a switch, which is controlled by the sample and hold control voltage $V_S$ and the capacitor C is used as a storage element. The analog voltage ($V_{in}$) to be sampled is applied to the drain and the sample and hold control voltage $V_S$ is applied to the gate terminal of E-MOSFET. The E-MOSFET conducts and acts as a closed switch during the positive portion of $V_S$ and this allows input voltage to charge capacitor. When sample and hold control voltage is zero, the E-MOSFET is OFF and acts as an open switch. Therefore, only the discharge path for capacitor is through operational amplifier. However, the input resistance of the operational amplifier voltage follower is also very high hence the voltage across the capacitor is retained. The time periods ($T_S$) of the sample and hold control voltage ($V_S$) during which the voltage across the capacitor is equal to the input voltage are called sample period. The time periods ($T_H$) of the sample and hold control voltage ($V_S$) during which the voltage across the capacitor is constant are called hold periods.

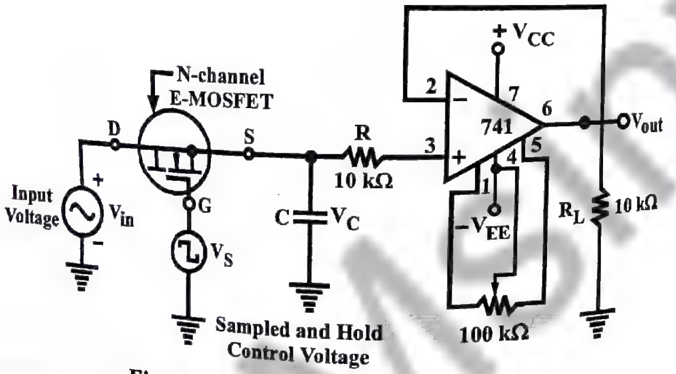


*Fig. 4.16 Sample and Hold Circuit*

Commonly the sample and hold circuit is used in digital interfacing and communications such as analog to digital systems. In a computer controlled system the sample and hold switch would be controlled by a digital signal from the computer.

***Q.30. What is the operating principle of voltage-to-frequency type ADC?***

*Ans.* In a voltage to frequency type ADC, pulses of frequency proportional to the analog voltage are generated and counted for a fixed time interval.

***Q.31. With the help of circuit diagram explain the V-F converter.***
***(R.G.P.V., June 2012)***

***Or***

***Explain the principle working of V-F converter. (R.G.P.V., Dec. 2012)***

***Or***

***With the help of circuit diagram explain the working of V-F converters.***
***(R.G.P.V., Dec. 2014)***

***Or***

***Explain voltage to frequency converter with the help of block diagram and waveforms. (R.G.P.V., June 2015)***

***Or***

***Write a short note on V-F converters.***
***(R.G.P.V., June 2009, 2010, Dec. 2015, June 2017, May 2018)***

*Ans.* An analog voltage can be converted into digital form by producing pulses, whose frequency is proportional to the analog voltage. These pulses are counted by a counter for a fixed duration and the reading of the counter will be proportional to the frequency of pulses and hence to the analog voltage.

**Operation** – The block diagram of voltage to frequency converter is shown in fig. 4.17. When an analog input voltage ($V_{in}$) is applied to an integrator, which in turn produces a ramp signal whose slope is proportional to the input voltage. When the output voltage ($V_o$) attains a certain value such as a present threshold level, a trigger pulse is produced and also a current pulse is generated, which is used to discharge the capacitor of integrator (C). Now here a new ramp is initiated. The time between successive threshold level crossing is inversely proportional to the slope of the ramp. Because the slope of the ramp is proportional to the input analog voltage ($V_{in}$), the frequency of the output pulses from the comparator is therefore directly proportional to the input analog voltage.

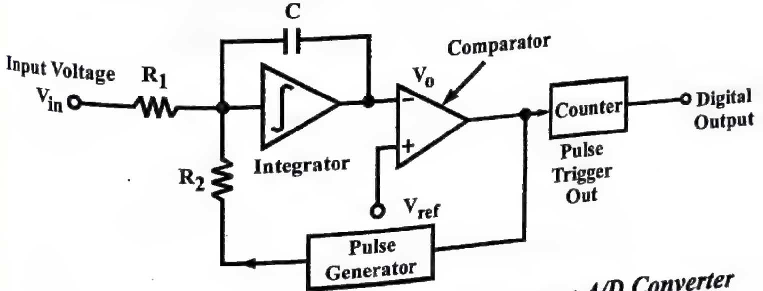


*Fig. 4.17 Block Diagram of Voltage to Frequency A/D Converter*

This output frequency can be measured with the help of a digital frequency counter. The accuracy of this method is comparable with the ramp type A/D converter and it is limited by the stability of the integrator time constant, the stability and the comparator accuracy.

The main applications of this type of converter is in noisy industrial environments, where small analog signals must be transmitted from transducer circuits to a control computer. The small analog signals can be strongly affected by noise if they are directly transmitted to the control computer. To feed the analog signal to a voltage controlled oscillator, which generates a digital signal whose output frequency changes according to the analog input. This digital signal is transmitted to the computer and it will be much less affected by noise.

*Q.32. Discuss V-F converter and D/A converter in short.*

***(R.G.P.V., June 2011)***

*Ans.* **V-F Converter** – Refer the ans. of Q.31.

**D/A Converter** – Refer the ans. of Q.17.

*Q.33. Explain in brief frequency-to-voltage converter.*

*Ans.* The block diagram of frequency to voltage converter is shown in fig. 4.18. The circuit charges the capacitor to a certain level. An integrator is connected in it and the capacitor discharges into this integrator or a low pass circuit. This happens for all the cycles of the input waveform. The precision switch and the monostable multivibrator generate a pulse of a specific amplitude and period which is fed into the averaging network. Hence we get a DC voltage at the output.

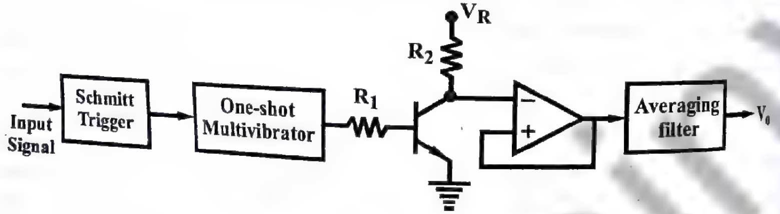


*Fig. 4.18*

## NUMERICAL PROBLEMS

***Prob.1. What is the resolution of a 12-bit D/A converter which uses a binary ladder ? If the full-scale output is + 10 V, what is the resolution in volts?***

***(R.G.P.V., Dec. 2011)***

*Sol.* The LSB in a 12-bit system has a weight of $\frac{1}{4096}$. Thus, this converter has a resolution of 1 part in 4096. The resolution expressed as a percentage is $\left(\frac{1}{4096}\right) \times 100 \approx$ 0.02%. The voltage resolution is determined by multiplying the weight of the LSB by the full-scale output voltage. Hence, the resolution in volts is $\left(\frac{1}{4096}\right) \times 10\text{ V} = \mathbf{2.44\ mV}$. **Ans.**

***Prob.2. An 8-bit D/A converter has a step size of 6 mV. Find full scale output voltage and percentage resolution.***

***[R.G.P.V., Dec. 2003(EL/ET), June 2014]***

*Sol.* (i) For 8-bit D/A converter there will be $2^8 – 1 = 255$ steps of size 6 mV each. The full scale output will be expressed as –

$$\text{Full scale output} = \text{Number of steps} \times \text{Step size}$$

$$= 255 \times 6\text{ mV} = \mathbf{1.53\ V} \qquad \textbf{Ans.}$$

$$\text{(ii) Percentage resolution} = \frac{\text{Step size}}{\text{Full scale output}} \times 100\%$$

or $$\text{Percentage resolution} = \frac{1}{\text{Total number of steps}} \times 100\%$$

$$= \frac{6\text{ mV}}{1.53\text{ V}} \times 100\% = \frac{1}{255} \times 100\% = \mathbf{0.39\%\,Ans.}$$

***Prob.3. Fig. 4.19 shows a computer control of motor speed. It can change motor speed from 0 to 1500 r.p.m. Find the number of bits of the computer so that it can control the speed within 1 r.p.m. of required speed.***

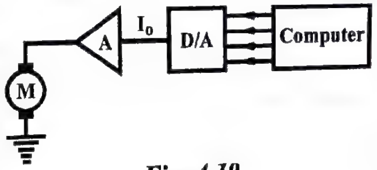


*Fig. 4.19*

***(R.G.P.V., June 2014)***

*Sol.* The motor speed will range from 0 to 1500 rpm as the DAC goes from zero to full scale. Each step in the DAC output will produce a step in the motor speed. We want the step size to be no greater than 1 r.p.m. Thus we need at least 1500 steps (1500/1). Now we must determine how many bits are required so that there are at least 1500 steps from zero to full scale.

We know that the number of steps is

$$2^N – 1$$

and so we can say

$$2^N – 1 \geq 1500$$

$$2^N \geq 1501$$

Since $2^{10} = 1024$ and $2^{11} = 2048$, the smallest number of bits that will produce at least 1500 steps is **11**. **Ans.**

***Prob.4. For a 5-bit resistive divider, determine the following –***

***(i) The weight assigned to the LSB***

***(ii) The weight assigned to the second and third LSB***

***(iii) The change in output voltage due to a change in the LSB, the second LSB and third LSB.***

***(iv) The output voltage for a digital input of 10101.***

***Assume 0 = 0V and 1 = + 10 V.***

***(R.G.P.V., Dec. 2010)***

***Sol.*** (i) The LSB weight is $1/(2^5 - 1) = \mathbf{1/31}$

(ii) The second LSB weight is **2/31** and the third LSB weight is 4/31. Ans.

(iii) The LSB causes a change in the output voltage of 10/31 V. The second LSB causes an output voltage change of 20/31 V, and the third LSB causes an output voltage change of 40/31 V. Ans.

(iv) The output voltage for a digital input of 10101 is Ans.

$$V_A = \frac{10\times2^0 + 0\times2^1 + 10\times2^2 + 0\times2^3 + 10\times2^4}{2^5 - 1}$$

$$= \frac{10(1+4+16)}{32-1} = \frac{210}{31} = \mathbf{+\,6.77\ V} \quad \text{Ans.}$$

***Prob.5. What is the resolution of a 9 bit D/A converter which uses a ladder network ? What is this resolution expressed as a percent ? If the full scale output voltage of this converter is +5V, what is the resolution in volts.***
***(R.G.P.V., Dec. 2008, 2013)***

***Sol.*** The LSB in a 9-bit system has a weight of 1/512. Thus this converter has a resolution of 1 part in 512. The resolution expressed in percentage is

$$= \frac{1}{512} \times 100 \text{ percent} \cong \mathbf{0.2\ percent} \quad \text{Ans.}$$

The voltage resolution is obtained by multiplying the weight of the LSB by the full-scale output voltage. Thus the resolution in volts is

$$= \frac{1}{512} \times 5 = \mathbf{9.8\ mV} \quad \text{Ans.}$$

## MULTIVIBRATORS – BISTABLE, MONOSTABLE, ASTABLE, SCHMITT TRIGGER, IC 555 & ITS APPLICATIONS

***Q.34. Explain linear waveshaping circuits.*** ***(R.G.P.V., Dec. 2013)***

***Ans.*** Linear waveshaping circuits generally depend on the two mathematical operations namely, differentiation and integration. These circuits alter the shape of a waveform by passing them through a process of differentiation or integration. A step waveform, as an example, can be converted into a decaying exponential waveform by transmitting it through a differentiator. The same step waveform when it passes through an integrator, generates a raising exponential waveform. This is the way new waveforms are produced from the existing waveforms.

***Q.35. What do you understand by multivibrator ?***

***Ans.*** Multivibrator is a switching circuit and may be defined as an electronic circuit that generates non-sinusoidal wave such as rectangular waves, sawtooth waves, square waves, etc. Multivibrators are capable of storing binary numbers, counting pulses, synchronising arithmetic operations and performing other essential functions as in digital systems.

There are three types of multivibrators –

(i) Bistable (ii) Monostable (iii) Astable

***Q.36. Explain the terms – monostable, bistable and astable multivibrator.***
***(R.G.P.V., Dec. 2015)***

***Ans.*** **Monostable Multivibrator** – Monostable multivibrator or one shot multivibrator, generates a single pulse of specified duration in response to each external trigger signal. As its name implies, only one stable state exists.

**Bistable Multivibrator** – Bistable multivibrator is also referred to as ***bistable latch*** or ***flip-flop.*** One important property of a bistable circuit is that it maintains a given output voltage level unless an external signal (trigger) is applied. Application of an appropriate external signal causes a change of state, and this output level is maintained indefinitely until a second trigger is applied. Thus, a bistable multivibrator requires two external triggers before it returns to its initial state.

**Astable Multivibrator** – Astable multivibrators also known as ***free running multivibrator***. The circuit has two quasistable states. Thus, there is oscillations between these two states and no external signal are required to produce the change in state. Astable circuits are used to generate square waves.

***Q.37. Explain the working of a bistable multivibrator.***
***(R.G.P.V., Dec. 2005, June 2009)***

***Or***

***Describe bistable multivibrator with diagram and working principle.***
***(R.G.P.V., June 2011)***

***Ans.*** Fig. 4.20 shows the circuit of a bistable multivibrator. It consists of two similar transistors $Q_1$ and $Q_2$ with equal collector resistors $R_{C1}$ and $R_{C2}$. The transistor $Q_1$ is coupled to the base of transistor $Q_2$ by resistor $R_1$ but the transistor $Q_2$ is coupled to the base of transistor $Q_1$ by resistor $R_2$. So the feedback is coupled through the resistors and not through the capacitors. The base resistors $R_3$ and $R_4$ are joined to common source ($-V_{BB}$). The resistors $R_1$ and $R_2$ are shunted by capacitors $C_1$ and $C_2$, respectively. These capacitors are known as commutating capacitors or speed up capacitors.

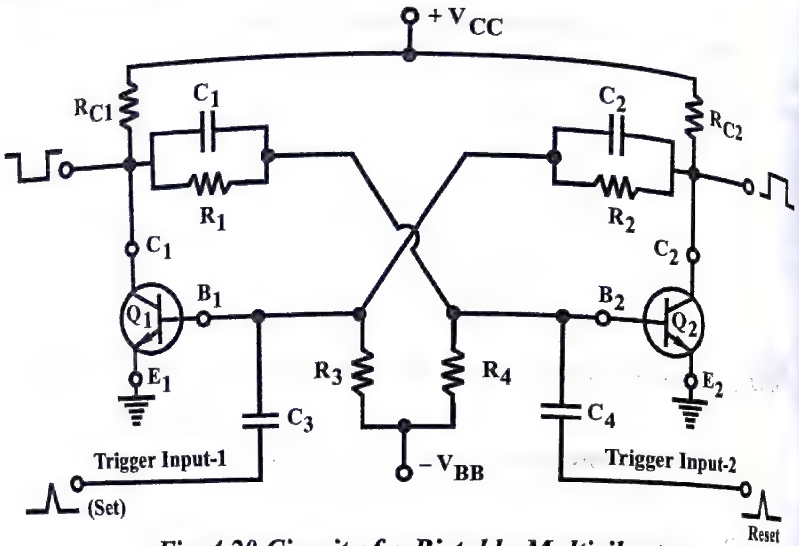


*Fig. 4.20 Circuit of a Bistable Multivibrator*

The time interval during which the conduction transfers from one transistor to another transistor is known as ***transition time***. The transition time is reduced by using small value capacitors in shunt with the coupling resistors. The output of a bistable multivibrator is available at the collector terminal of both the transistors $Q_1$ and $Q_2$. However, the two outputs are complements of each other.

**Working** – When the $V_{CC}$ supply is switched-on transistor $Q_1$ will start conducting more than transistor $Q_2$. Because of the feedback action, transistor $Q_1$ turns-on while the transistor $Q_2$ is cut-off. This is the first stable state and will remain in this state till a trigger input is applied at the base of transistor $Q_1$. A negative pulse of sufficient magnitude applied to set input will turn-off the transistor $Q_1$ and $Q_2$ to turn-on. Let a positive pulse is applied at the base of transistor $Q_2$ through capacitor $C_4$. It will cause the transistor $Q_2$ to conduct. As the collector voltage of transistor $Q_2$ falls, it cuts-off the transistor $Q_1$. Thus the circuit switches to a state where transistor $Q_1$ turns-off and $Q_2$ turns-on. The output waveform of bistable multivibrator is shown in fig. 4.21.

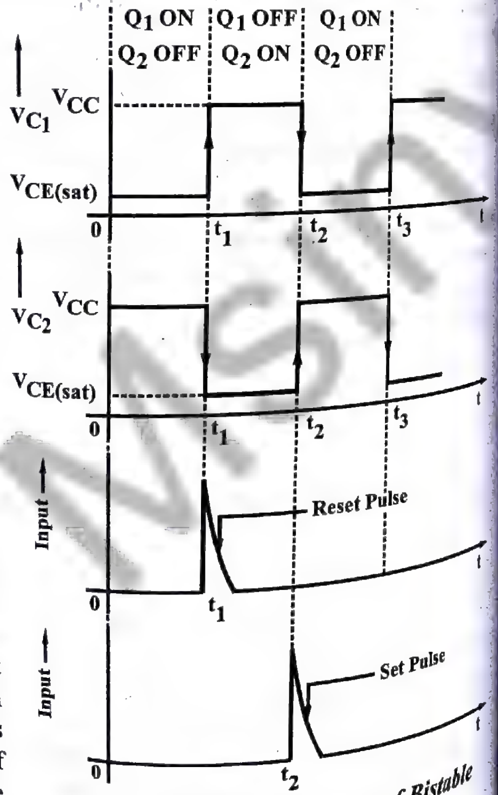


*Fig. 4.21 Output Waveforms of Bistable Multivibrator*

Now if a positive pulse is applied at transistor $Q_1$ and a negative pulse at transistor $Q_2$, it will switch the circuit back to its original stable state, i.e., transistor $Q_1$ turns-on and $Q_2$ turns-off. From the waveforms in fig. 4.21 it is noted that the output waveforms are the complement of each other.

***Q.38. Write down the applications of bistable multivibrator.***

***Ans.*** The applications of bistable multivibrator are as follows –

(i) It is used as memory element in shift registers, counters and so on.

(ii) It can also be used as a frequency divider.

(iii) It is used to generate square waves of symmetrical shape by sending regular triggering pulse to the input.

***Q.39. Explain the operation of monostable multivibrator with the help of necessary diagrams and waveforms. Describe the applications of it.***
***(R.G.P.V., Dec. 2010)***

***Or***

***Explain the working of monostable multivibrators with the help of waveforms and circuit diagram.*** ***(R.G.P.V., June 2010)***

***Or***

***With the help of timing diagram explain the working of monostable multivibrator.*** ***(R.G.P.V., June 2012)***

***Or***

***With the help of circuit diagram and timing waveforms explain the working of monostable multivibrator.*** ***(R.G.P.V., Dec. 2014)***

***Or***

***Explain the operation of monostable multivibrator with the help of waveforms.*** ***(R.G.P.V., June 2015)***

***Or***

***Explain monostable multivibrator and write its applications.***
***(R.G.P.V., Dec. 2017)***

***Or***

***Draw and explain the working of monostable multivibrator.***
***(R.G.P.V., May 2018)***

***Ans.*** Fig. 4.22 shows the circuit of a monostable multivibrator. It consists of similar transistors $Q_1$ and $Q_2$ with equal collector resistances, (i.e., $R_{C1} = R_{C2}$). The output of transistor $Q_1$ is coupled to the base of transistor $Q_2$

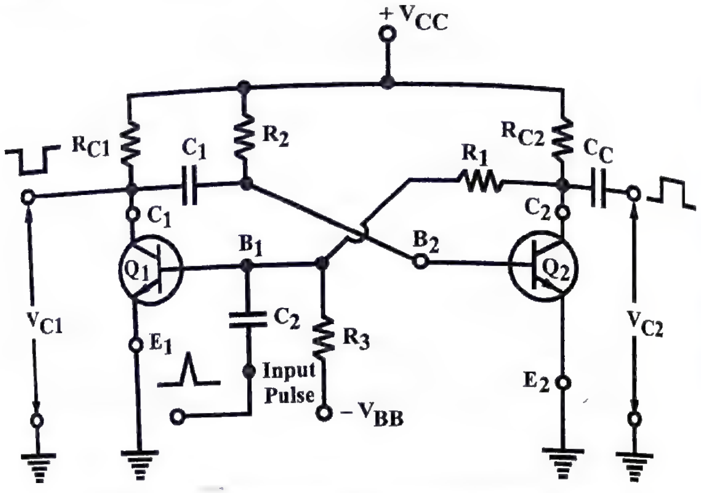


*Fig. 4.22 Monostable Multivibrator Circuit*

through capacitor $C_1$. The output of transistor $Q_2$ is coupled to base of transistor $Q_1$ through resistor $R_1$. The base of transistor $Q_2$ is returned to the $V_{CC}$ supply through a resistor $R_2$, while the base of transistor $Q_1$ is connected to the negative supply through a resistor $R_3$. The input pulse is applied through capacitor $C_2$. The output may be taken from either transistor $Q_1$ or $Q_2$. However, the two outputs are the complement of each other, i.e., if one of the output is at $V_{CC}$ level, the other is at $V_{CE(sat)}$ level.

**Working** – When the $V_{CC}$ supply is switched-on but with no input pulse is applied, transistor $Q_2$ is 'ON' and is operating in its saturation region. It is being forward biased by $V_{CC}$ supply and base resistor $R_2$. The collector of transistor $Q_2$ is virtually at ground potential. Because of the forward bias base-emitter junction, the base of the transistor $Q_2$ is 0.7 V above ground potential. The transistor $Q_1$ is cut-off being reverse biased by supply $V_{BB}$ and resistor $R_3$. Its base resistor $R_1$ is connected to the collector of transistor $Q_2$ which is a zero potential. Thus collector resistor $R_{C1}$ is completely disconnected from the ground emitter of transistor $Q_1$ and is free to carry current to charge capacitor $C_1$. The waveforms at the base and collector of the transistors $Q_1$ and $Q_2$ of monostable multivibrator are shown in fig. 4.23.

When a positive trigger pulse of short period and sufficient magnitude is applied to the base of transistor $Q_1$ through capacitor $C_2$, it overrides the reverse bias of emitter-base junction of transistor $Q_1$ and gives it a forward bias. Thus transistor $Q_1$ starts conducting. The collector voltage of transistor $Q_1$ falls due to the voltage drop across resistor $R_{C1}$. This fall in voltage is coupled through capacitor $C_2$ which decreases the forward bias of transistor $Q_2$. Because of the reduced forward bias, the collector current of transistor

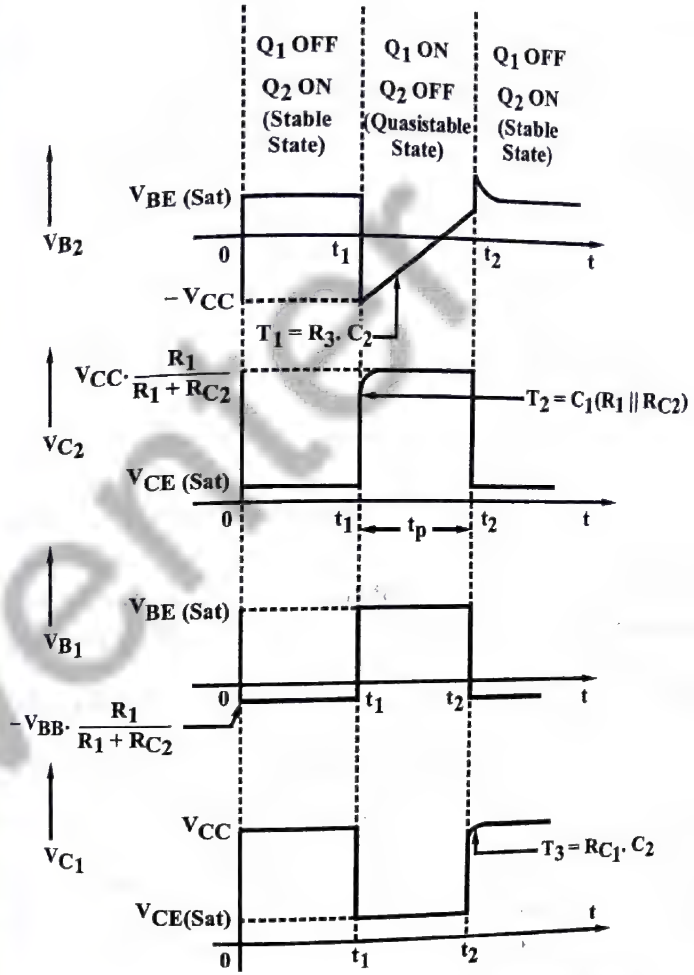


*Fig. 4.23 Waveforms for Monostable Multivibrator*

$Q_2$ starts decreasing and its collector voltage rises exponentially towards $V_{CC}$ $[R_1/(R_1 + R_{C2})]$ with a time constant $T_2 = C_1 (R_1 \parallel R_{C2})$.

The rising collector voltage of transistor $Q_2$ is coupled to the base of the transistor $Q_1$ through resistor $R_1$, where it further increases its forward bias. Because of increased forward bias, the transistor $Q_1$ conducts more. This is cummulative action because of the positive feedback and the collector voltage of transistor $Q_1$ falls to $V_{CE\ (sat)}$ level. The capacitor $C_1$ immediately starts charging towards $V_{CC}$ supply with a time constant $T_1 = R_3C_2$. As capacitor $C_1$ charges, the voltage at the base of the transistor $Q_2$ decreases. The transistor $Q_2$ is also pulled out from the cut-off and the reverse transition takes place, i.e., the transistor $Q_2$ turns-on and transistor $Q_1$ turns-off.

If the transistor $Q_2$ starts conducting, its collector voltage falls because of the voltage drop across collector resistor $R_{C2}$. This voltage drop is coupled to the base of transistor $Q_1$ whose collector voltage rises towards $V_{CC}$ supply with time constant $T_3 = R_{C1} C_1$. Finally the transistor $Q_2$ turns-on and transistor $Q_1$ goes 'OFF'. The circuit remains in stable state till another pulse is applied.

**Applications** – The applications of monostable multivibrator are as follows–

(i) The monostable multivibrator is used to function as an adjustable pulse width generator.

(ii) It is used to generate clean and sharp pulses from the distorted pulses.

(iii) It is used to generate uniform width pulses from a variable width input pulse train.

(iv) It is used as a time delay unit since it produces a transition at a fixed time after the trigger signal.

***Q.40. With the help of circuit diagram explain the working of astable multivibrator.*** ***(R.G.P.V., Dec. 2012)***

***Or***

***Explain astable multivibrator.*** ***(R.G.P.V., Dec. 2013)***

***Or***

***Write short note on astable multivibrator.*** ***(R.G.P.V., June 2008)***

***Ans.*** The circuit of an astable multivibrator using two similar transistors is depicted in fig. 4.24.

This circuit consists of two CE amplifier stages, each providing a feedback to the other. The feedback ratio is unity and positive because of 180° phase shift in each amplifier stage. Here the output of transistor $Q_1$ is coupled to the input of transistor $Q_2$ through capacitor $C_1$ but the output of transistor $Q_2$ is coupled to the input of transistor $Q_1$ through capacitor $C_2$. During the saturation region, the resistors $R_1$ and $R_2$ provide ON state base current to both the transistors respectively. In a symmetrical multivibrator $R_{C1} = R_{C2}$, $R_1 = R_2 = R$ and $C_1 = C_2 = C$.

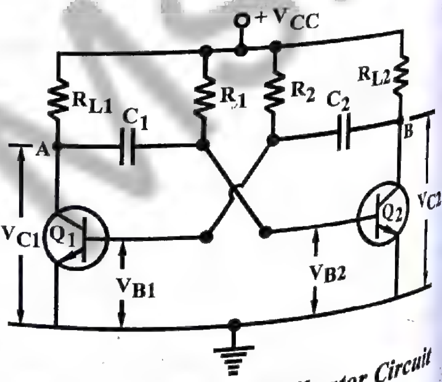


***Fig. 4.24 Astable Multivibrator Circuit***

The output can be taken from point A or B. There would be a phase reversal among the two outputs.

**Working** – When the $V_{CC}$ supply is switched-on, the transistor $Q_1$ conducts more than transistor $Q_2$ due to some imbalance in the circuit. The rising collector current in transistor $Q_1$ drives its collector more and more negative. This is applied to the base of transistor $Q_2$ through capacitor $C_1$ and establishes a reverse bias on transistor $Q_2$ thus its collector current starts decreasing. This is connected to the base of transistor $Q_1$ through capacitor $C_2$. Thus the transistor $Q_1$ is more forward bias. The further increase in collector current of transistor $Q_1$ causes a further decrease of collector current of transistor $Q_2$. Thus at $t > 0$, the transistor $Q_1$ is 'ON' and $Q_2$ is 'OFF'. In this position $V_{B1} = V_{BE\,(sat)}$, $V_{C1} = V_{CE(sat)}$, $V_{B2}$ is negative and $V_{C2} = V_{CC}$. The waveforms at base and collector of transistors $Q_1$ and $Q_2$ in astable multivibrator mode are shown in fig. 4.25.

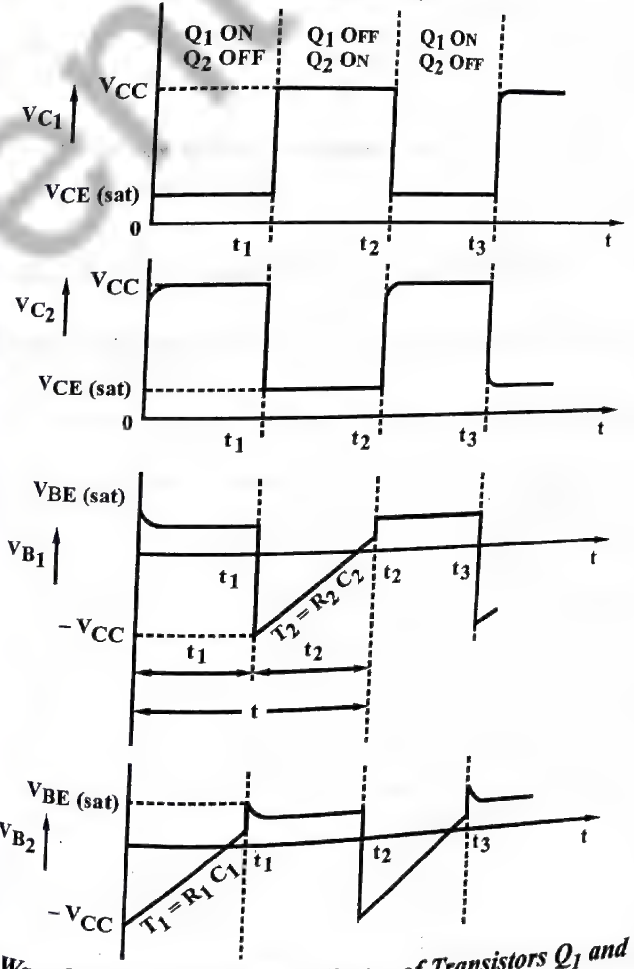


***Fig. 4.25 Waveforms at Base and Collector of Transistors $Q_1$ and $Q_2$ in Astable Multivibrator***

When transistor $Q_1$ is 'ON' and transistor $Q_2$ is 'OFF', the capacitor $C_1$ starts charging through resistor $R_1$. The charging takes place exponentially with time constant $T_1 = R_1 C_1$. When the voltage across capacitor $C_1$ becomes more than 0.7 V, it forward biases the transistor $Q_2$. So the transistor $Q_2$ starts conducting.

When transistor $Q_2$ is conducting, the potential of point B decreases. This is applied to the base of transistor $Q_1$ through capacitor $C_2$. As a result, the transistor $Q_2$ is pulled out of saturation. After few cycles, the transistor $Q_2$ is driven into saturation but point A is at $V_{CC}$. Now the capacitor $C_2$ starts charging through resistor $R_2$. When the voltage across capacitor $C_2$ becomes more than 0.7 V, it forward biases the transistor $Q_1$. So the transistor $Q_1$ starts conducting. The whole cycle is repeated again.

***Q.41. Enlist the applications of astable multivibrator.***

***Ans.*** The applications of astable multivibrator are as follows –

(i) It is used as square wave generator, voltage to frequency converter and in pulse synchronization and so on.

(ii) It is used in the construction of digital voltmeter and SMPS.

(iii) It can be operated as an oscillator over a wide range of audio and radio frequencies.

***Q.42. Explain the difference between various types of multivibrators.***

***Or***

***Distinguish between monostable and astable multivibrator.*** ***(R.G.P.V., June 2017)***

***Ans.*** The difference between various types of multivibrators are as follows –

| *S. No.* | *Astable Multivibrator* | *Bistable Multivibrator* | *Monostable Multivibrator* |
|---|---|---|---|
| (i) | The circuit has no stable states. | It is also referred to as bistable latch. | Only one stable state exists. |
| (ii) | No external signal is required to produce the change in state. | Application of a trigger signal causes a change of state. | Application of a trigger signal causes a change to the quasistable state. |
| (iii) | Astable multivibrator circuits are used to generate square waves. | Bistable multivibrators are used for counting and storing binary information in computer circuits. | The monostable multivibrator can be used as a frequency divider by adjusting the length of the timing cycle with respect to the time period T of the trigger input signal. |
| (iv) | It is used in the construction of digital voltmeter and SMPS. | It can also be used as a frequency divider. | It is used to generate clean and sharp pulses from the distorted pulses. |
| (v) | It can be operated as an oscillator over a wide range of audio and radio frequencies. | It is used to generate square waves of symmetrical shape by sending regular triggering pulse to the input. | It is used as a time delay unit since it produces a transition at a fixed time after the trigger signal. |

***Q.43. Discuss Schmitt trigger circuits. (R.G.P.V., June 2009, Dec. 2013)***

***Or***

***Write short note on Schmitt trigger.*** ***(R.G.P.V., June 2010, 2012)***

***Or***

***Draw a Schmitt trigger circuit and explain with waveforms.***
***(R.G.P.V., Nov./Dec. 2007, June 2015)***

***Or***

***With the help of circuit diagram explain the working of Schmitt trigger.***
***(R.G.P.V., Dec. 2012, 2014)***

***Or***

***Draw the circuit diagram of Schmitt trigger and explain its working.***
***(R.G.P.V., Dec. 2015)***

***Or***

***What is Schmitt trigger circuit ?*** ***(R.G.P.V., June 2014)***

***Or***

***Write short note on Schmitt trigger circuits.*** ***(R.G.P.V., Dec. 2017)***

***Ans.*** An inverting comparator with positive feedback which converts an irregular shaped waveform to a square wave is shown in fig. 4.26. This circuit is called as the Schmitt trigger circuit. The input voltage ($v_{in}$) triggers the output voltage $v_o$ every time it exceeds certain voltage levels known as upper threshold voltage ($V_{UT}$) and the lower threshold voltage ($V_{LT}$) as shown in fig. 4.26. These threshold voltages are obtained by using the voltage divider method between resistors $R_1$ and $R_2$. Where the voltage across resistor $R_1$ is fed back to the positive input. The voltage across resistor $R_1$ is a variable reference threshold voltage, which depends on the value and the polarity of the output voltage $v_o$. When the output voltage $v_o = +V_{sat}$, the voltage across resistor $R_1$

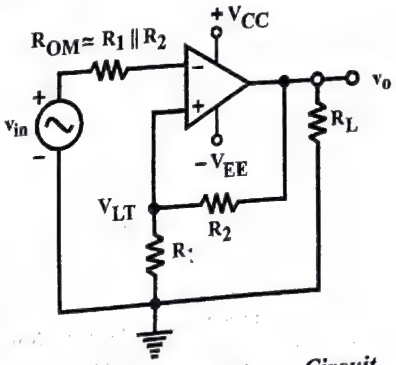


*Fig. 4.26 Schmitt Trigger Circuit*

is called the upper threshold voltage ($V_{UT}$). The input voltage ($v_{in}$) must be slightly more positive than upper threshold voltage ($V_{UT}$) in order to cause the output ($v_o$) to switch from ($+V_{sat}$) to ($-V_{sat}$). As long as the input voltage ($v_{in}$) is less than upper threshold voltage ($V_{UT}$) then the output voltage ($v_o$) is at ($+V_{sat}$). By using the voltage divider method we obtain –

$$V_{UT} = \frac{R_1}{R_1 + R_2}(+V_{sat}) \quad ...(i)$$

When the value of output voltage $v_o = -V_{sat}$, the voltage across resistor $R_1$ is called lower threshold voltage ($V_{LT}$). The input voltage ($v_{in}$) must be slightly more negative than the lower threshold ($V_{LT}$) in order to cause output voltage ($v_o$) to switch from ($-V_{sat}$) to ($+V_{sat}$). For the input voltage ($v_{in}$) values greater than the lower threshold voltage ($V_{LT}$) then the output voltage is at ($-V_{sat}$). The lower threshold voltage is given by

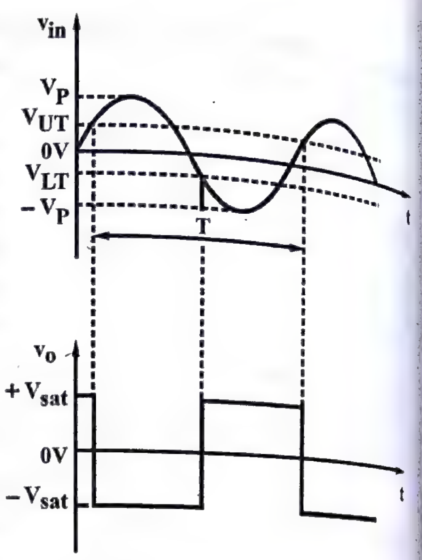


*Fig. 4.27 Input and Output Waveforms of the Schmitt Trigger Circuit*

$$V_{LT} = \frac{R_1}{R_1 + R_2}(-V_{sat}) \quad ...(ii)$$

The resistance $R_{OM} \cong R_1 \| R_2$ is used to minimize the offset problems. If the input is a sine wave then the output of the Schmitt trigger is a square wave, as shown in fig. 4.27. If the input of the comparator increased upto the upper threshold voltage level then its output switches from ($+V_{sat}$) to ($-V_{sat}$) and reverts back to its original state, ($+V_{sat}$) when the input goes below lower threshold voltage ($V_{LT}$). The hysteresis voltage is equal to the difference between the upper threshold voltage and lower threshold voltage.

Therefore,

$$V_{HY} = V_{UT} - V_{LT}$$

$$= \frac{R_1}{R_1 + R_2}[(+V_{sat}) - (-V_{sat})] \quad ...(iii)$$

The output voltage versus input voltage plot of the hysteresis voltage is shown in fig. 4.28.

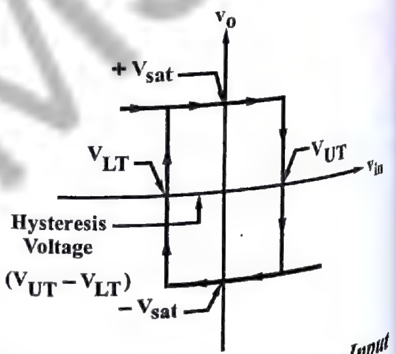


*Fig. 4.28 Output Voltage versus Input Voltage Plot of the Hysteresis Voltage*

### *Q.44. Sketch circuit of Schmitt trigger and explain its operation. What is hysteresis ?* *(R.G.P.V., Feb. 2010)*

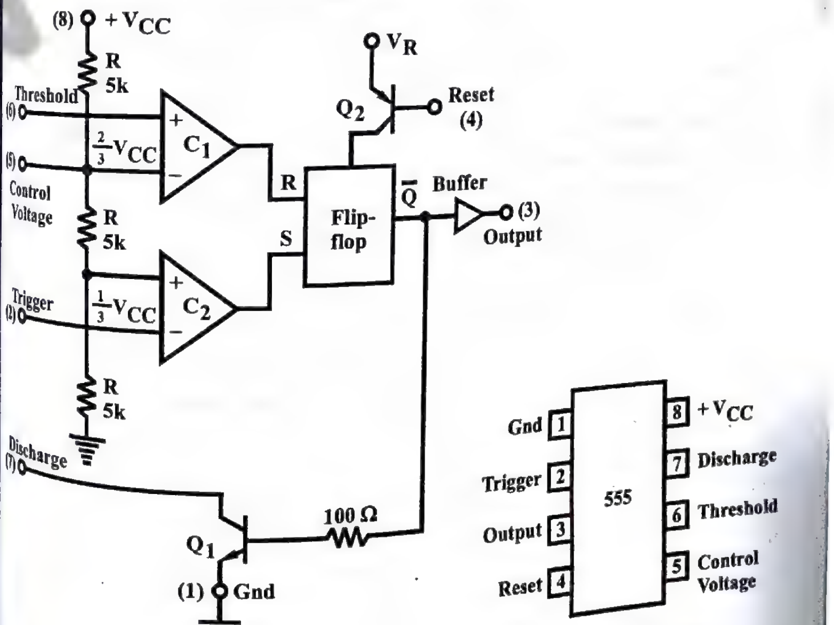
*Ans.* **For Schmitt Trigger** – Refer the ans. of Q.43.

**Hysteresis** – The lagging of the lower threshold voltage from the upper threshold voltage is known as the hysteresis.

### *Q.45. Write about 555 timer and draw its functional and pin diagram. Also write applications of them.*

*Ans.* The astable and monostable circuits are so commonly required that special monolithic ICs, called ICs timers, have been made available. The timer 555, is one example which has gained wide acceptance in terms of cost and versatility. It was first introduced by Signetic corporation as SE/NE 555.

The device 555 is a monolithic timing circuit that can produce accurate and highly stable time delays or oscillations. Like general purpose Op-Amp, the 555 timer is reliable, easy to use, and economical. The timer 555 is available as an 8 pin metal can, an 8 pin mini DIP, or a 14 pin DIP. Fig. 4.29 shows the functional diagram and the pin configuration of the SE/NE 555 timer. The SE 555 is designed for the operating temperature range from – 55° to + 125°C, while the NE 555 operates over a temperature range of 0° to 70°C.



*(a) Functional Diagram*

*(b) Pin Configuration*

*Fig. 4.29 The Timer 555*

The important features of the NE 555 timer are as follows –

(i) + 5 to + 18 V supply voltage

(ii) Adjustable duty cycle

(iii) Timing from microsecond to hours

(iv) High current output

(v) Capacity to source or sink current of 200 mA

(vi) Output can drive TTL

(vii) Temperature stability of 50 parts per million per °C change in temperature or 0.005% per °C

(viii) Reliable, easy to use, and low cost like general purpose op-amp.

**Pin Configuration Description –**

**Pin 1 (Ground)** – All voltages are measured with respect to this terminal.

**Pin 2 (Trigger)** – The output of the timer depends on the amplitude of the external trigger pulse applied to this pin. This is explained above.

**Pin 3 (Output)** – There are two ways a load can be connected to the output terminal either between pin 3 and pin 1 or between pin 3 and supply voltage + $V_{CC}$ (pin 8).

**Pin 4 (Reset)** – The device 555 is reset by applying a negative pulse to this pin when the reset function is not in use, the reset terminal should be connected to + $V_{CC}$ to avoid any possibility of false triggering.

**Pin 5 (Control Voltage)** – An external voltage applied to this terminal changes the threshold as well as the trigger voltage. In other words, by imposing a voltage on this pin or by connecting a potentiometer between this pin and ground, the pulse width of the output waveform can be varied. When not used, the control pin should be bypassed to ground with a 0.01 μF capacitor to prevent any noise disturbances.

**Pin 6 (Threshold)** – This is the non-inverting terminal of comparator $C_1$, which monitors the voltage across the external capacitor.

**Pin 7 (Discharge)** – This pin is connected internally to the collector of transistor $Q_1$ as shown in fig. 4.29 (a).

**Pin 8 (+ $V_{CC}$)** – The supply voltage of + 5 V to + 18 V is applied to this pin with respect to ground (pin 1).

**Applications of 555 Timer** – Timer 555 is used in number of novel and useful applications. Some typical applications are given below –

(i) Monostable and astable multivibrators

(ii) D.C.-D.C. converters

(iii) Digital logic probes

(iv) Waveform generators

(v) Analog frequency meters and tachometers

(vi) Temperature measurement and control

(vii) Infrared transmitter

(viii) Burglar and toxic gas alarms

(ix) Voltage regulator.

***Q.46. Describe the monostable multivibrator using IC 555 timer.***
***(R.G.P.V., Dec. 2008)***

*Ans.* The monostable multivibrator using 555 timer is shown in fig. 4.30. In the stand by state, referring to fig. 4.31 (a), the control flip-flop holds $Q_1$ ON, thus clamping the external timing capacitor C to ground. The output (pin 3) during this time is at ground potential, or LOW. The three 5 kΩ internal resistors act as voltage dividers providing bias voltages of (2/3) $V_{CC}$ and (1/3) $V_{CC}$ respectively. Since these two voltages fix the necessary comparator threshold voltages, they also aid in determining the timing interval.

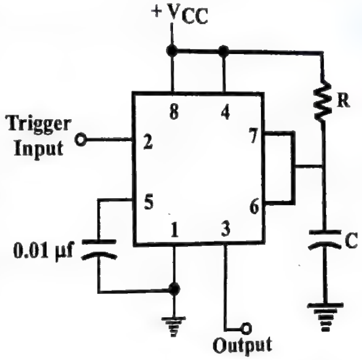
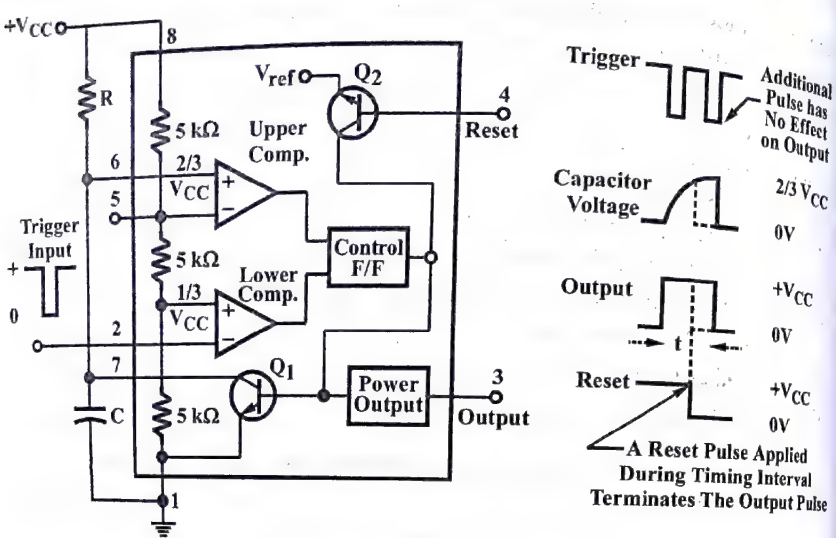


*Fig. 4.30 Monostable Multivibrator using IC 555 Timer*

Since the "lower" comparator is biased at (1/3)$V_{CC}$, it remains in the standby state so long as the trigger input (pin 2) is held above (1/3)$V_{CC}$. When triggered only by a negative going pulse, the lower comparator sets the internal flip-flop which releases the short circuit across the timing capacitor thus turning $Q_1$ OFF and the output goes HIGH (approximately equal to $V_{CC}$). Since the timing capacitor is now unclamped, the voltage across it now rises exponentially through R towards $V_{CC}$, with a time constant RC. After a period of time, the capacitor voltage will equal (2/3)$V_{CC}$ and the "upper" comparator resets the internal flip-flop, which in turn discharges the capacitor rapidly to ground potential, turning $Q_1$ ON. As a consequence, the output now returns to the standby state or ground.



*(a) Block Diagram* *(b) Timing Pulses of Monostable Multivibrator using IC 555 Timer*

*Fig. 4.31*

The 555 monostable timing sequence is shown in fig. 4.31 (b). The circuit triggers only on a negative going pulse when the level is less than (1/3) $V_{CC}$. Once triggered, the output will remain HIGH until the set time has elapsed, even if it is triggered again during this interval. Since the external capacitor voltage changes exponentially from 0 to (2/3) $V_{CC}$.

$$\Delta V = V_{CC}(1 - e^{-t/RC})$$

$$\frac{2}{3}V_{CC} = V_{CC}(1 - e^{-t/RC})$$

or

$$t = -RC\, ln\left(\frac{1}{3}\right)$$

When the output is HIGH, the time interval becomes

$$t = 1.1\ RC\ \text{(seconds)}.$$

***Q.47. Describe the astable multivibrator using IC 555 timer.***

***(R.G.P.V., Dec. 2008)***

***Or***

***Implement an astable multivibrator using 555 timer IC.***

***(R.G.P.V., June 2014)***

***Ans.*** The monolithic integrated circuit 555 timer can be used as square wave generator, linear saw-tooth generator, pulse generator, time delay generator etc. The 555 timer circuit to generate square waveform (in astable mode) and trian-gular waveform is shown in fig. 4.32. The capacitor $C_t$ charges through the resistances $R_1$ and $R_2$ and discharges through the resistance $R_2$ only. The duty cycle can be controlled by the ratio of resistances $R_1$ and $R_2$. Here the capacitor charges and discharges between (1/3) $V_{CC}$ and (2/3) $V_{CC}$. The charging and discharging times are independent of supply voltage.

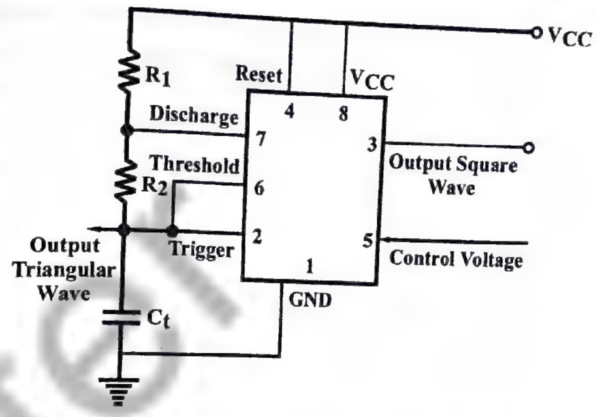


***Fig. 4.32 Square and Triangular Waveforms Generator using 555 Timer***

The charging time of the capacitor is

$$t_1 = 0.693\ (R_1 + R_2)\ C_t$$

The discharging time of the capacitor is

$$t_2 = 0.693\ R_2\ C_t$$

Therefore, the time period of the waveform is given by

$$T_1 = t_1 + t_2 = 0.693\ (R_1 + 2R_2)\ C_t$$

The frequency of the oscillation is given by

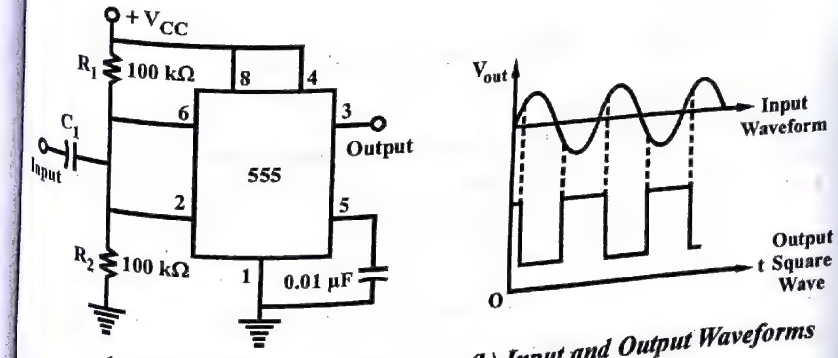
$$f = 1/T = 1.44/C_t\ (R_1 + 2R_2)$$

where $R_1$ and $R_2$ are in ohms and $C_t$ is in farads.

The 555 timer can also be used for the construction of monostable multivibrator and Schmitt trigger.

***Q.48. Describe the Schmitt trigger using IC 555 timer.***

***Ans.*** The IC 555 timer can be used to function as a variable threshold Schmitt trigger. Since the internal circuitry has a high input impedance and



*(a) Circuit Diagram* *(b) Input and Output Waveforms*

***Fig. 4.33 Schmitt Trigger using IC 555 Timer***

latching capability, the threshold voltage can be adjusted over a wide range with simultaneous open-collector/totempole outputs.

Schmitt trigger circuit is shown in fig. 4.33 (a) where the two internal comparator inputs (pins 2 and 6) are connected together and externally biased at (1/2) $V_{CC}$ through $R_1$ and $R_2$. Since the upper comparator at pin 6 will trip at (2/3)$V_{CC}$ and the lower comparator at (1/3) $V_{CC}$ the bias provided by $R_1$ and $R_2$ is centered within these two thresholds.

A sinewave input of sufficient amplitude to exceed the reference levels causes the internal flip-flop to alternatively set and reset, generating a square wave output. As long as $R_1$ equals $R_2$, the 555 timer will automatically be biased for any supply voltage in the range of 5 to 16 V. From the curve shown in fig. 4.33 (b), it is observed that there is a 180° phase shift.

Unlike a conventional multivibrator type of square wave generator that divides the input frequency by 2, the main advantage of schmitt trigger is that it simply converts the sinewave signal without division.

## NUMERICAL PROBLEMS

***Prob.6. In an astable multivibrator, the base resistor are of 12.5 kΩ and the capacitors are of 0.01 μF. Determine the PRR (pulse repetition rate).* (R.G.P.V., Feb. 2010)**

***Sol.*** Given, R = 12.5 kΩ, C = 0.01 μF

We know that

The period of oscillation

$$T = \frac{1}{f} = 1.386\ RC$$

where f = Pulse repetition rate

$$f = \frac{1}{1.386\ RC} = \frac{1}{1.386 \times 12.5 \times 10^3 \times 0.01 \times 10^{-6}} = 5772\ \text{Hz}$$

$$= \mathbf{5.772\ kHz}$$ **Ans.**

***Prob.7. Determine the frequency of oscillation for the free running multivibrator circuit shown below. It is given that $R_A = R_B = 1\ k\Omega$ and $C = 1000\ pF$. Also calculate the duty cycle.* (R.G.P.V., Dec. 2010)**

***Sol.*** Given that $R_A = R_B = 1\ k\Omega = 1000\ \Omega$

$C = 1000\ pF = 1000 \times 10^{-12} = 1 \times 10^{-9}$

We know that $T_1 = 0.7C\ (R_A + R_B)$

$= 0.7 \times 1 \times 10^{-9}\ (1000 + 1000) = 1.4 \times 10^{-6} = 1.4\ \mu s$

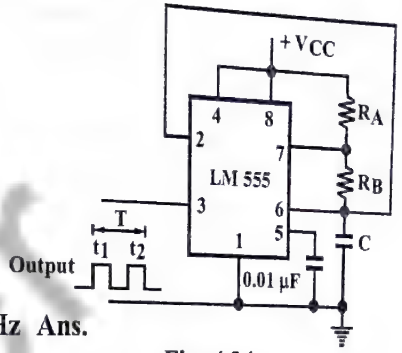


*Fig. 4.34*

$T_2 = 0.7\ CR_B$

$= 0.7 \times 1 \times 10^{-9} \times 1000$

$= 0.7 \times 10^{-6} = 0.7\ \mu s$

$T = T_1 + T_2$

$= (1.4 + 0.7)\ \mu s$

$= 2.1\ \mu s$

Frequency of oscillations,

$$f = \frac{1}{T} = \frac{1}{2.1 \times 10^{-6}} = \mathbf{476\ kHz}$$ **Ans.**

Duty cycle,

$$D = \frac{R_A + R_B}{R_A + 2R_B} \times 100 = \frac{(1000 + 1000)}{(1000 + 2000)} \times 100 = \mathbf{66.67\%}$$ **Ans.**

## TTL, PMOS, CMOS AND NMOS LOGIC, INTERFACING BETWEEN TTL TO MOS

***Q.49. What do you understand by logic families ?* (R.G.P.V., June 2015)**

***Ans.*** Basically, there are two types of semiconductor devices namely bipolar and unipolar. Based on these devices, digital integrated circuits have been made which are commercially available. Various digital functions are being fabricated in a variety of forms using bipolar and unipolar technologies. A group of compatible ICs with the same logic levels and supply voltages for performing various logic functions have been fabricated using a specific circuit configuration. This configuration is known as a ***logic family***.

***Q.50. Write characteristics of digital logic families.* (R.G.P.V., Dec. 2015)**

***Ans.*** Some characteristics of digital logic families are as follows –

(i) Speed of operation (propogation delay)
(ii) Power dissipation
(iii) Figure of merit
(iv) Fan-in
(v) Fan-out
(vi) Current and voltage parameters
(vii) Noise immunity (noise margin)
(viii) Operating temperature
(ix) Power supply requirements
(x) Flexibilities available.

***Q.51. Give classification of logic families. Also list the characteristics of digital IC.* (R.G.P.V., Dec. 2016)**

***Ans.*** **Classification** – The various logic families can be placed into two broad categories according to the IC fabrication process are given below –

*(i) **Bipolar Logic Families*** – Based on the two main operations of bipolar ICs, bipolar families are classified into saturated logic and non-saturated logic.

**(a) Saturated Bipolar Logic Families –**

(1) Resistor-transistor logic (RTL)

(2) Direct-coupled transistor logic (DCTL)

(3) Diode-transistor logic (DTL)

(4) High threshold logic (HTL)

(5) Transistor-transistor logic (TTL or $T^2L$)

(6) Integrated-injection logic ($I^2L$).

**(b) Non-saturated Logic Families –**

(1) Schottky TTL (2) Emitter-coupled logic (ECL).

*(ii) **MOS Logic Families*** – The MOS families include following –

(a) PMOS p channel MOSFETs

(b) NMOS n channel MOSFETs

(c) CMOS complementary MOSFETs.

**Characteristics** – Refer the ans. of Q.50.

***Q.52. Explain the operation of the RTL NOR gate.***

***(R.G.P.V., June 2008)***

***Ans.*** Fig. 4.35 shows the basic diagram of RTL NOR gate. The basic circuit of the RTL digital logic family is the NOR gate. Each input is associated with one resistor and one transistor. The collectors of the transistors are tied together at the output. A two-input RTL NOR gate driving n-similar gates is shown in fig. 4.35, which can be extended to accommodate a larger number of inputs.

The inputs representing the logic levels are applied at the terminals A and B. The voltage corresponding to low level should be low enough to drive the corresponding transistor to cut-off. In a similar manner, the input voltage corresponding to high level should be high enough to drive the corresponding transistor to saturation.

If both the inputs are low (or logic-0), transistors $Q_1$ and $Q_2$ are cut-off and the output is high (or logic-1). A high level on any input will drive the corresponding transistor to saturation causing the output to go low. The low level output voltage is $V_{CE\,(sat)}$ of a transistor ($\simeq$0.2V) and the high level output voltage depends on the number of gates connected to the output. This results the output voltage to be variable and is a deciding factor for the fan-out of the gate.

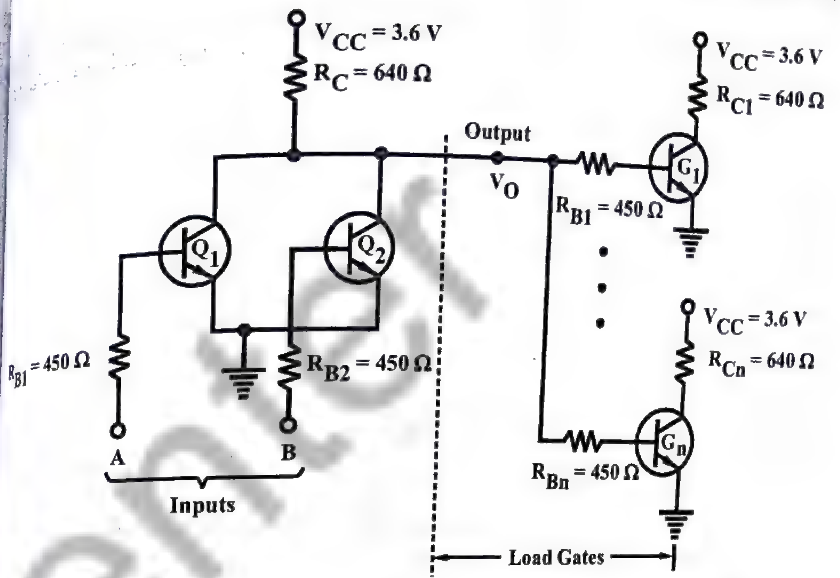


***Fig. 4.35 Two-input RTL NOR Gate Driving 'n' Similar Gates***

The base current is practically independent of the emitter junction characteristic. The resistor increases the input resistance and reduces the switching speed of the circuit. This degrades the rise and fall times of any input pulse. If all the inputs to the gate are low, the output is high and if the gate is not driving any other gate, that is, no load is connected, the output voltage will be slightly less than $V_{CC}$ (there is voltage drop across the common collector resistor due to $I_{CO}$ of transistors $Q_1$ and $Q_2$). If n similar gates are being driven, the load will be equivalent to a resistor of value 450/n ohms in series with a voltage source of 0.8 V. The base current for each load transistor is illustrated as –

$$I_B = \left[\frac{V_{CC}-0.8}{\left(R_C+\dfrac{450}{n}\right)}\right]\times\frac{1}{n} = \frac{2.8}{640n+450}$$

The collector current for the load transistor in saturation is given by

$$I_{C(sat)} = \frac{V_{CC}-0.2}{R_C} = \frac{3.6-0.2}{640} = 5.31\,\text{mA}$$

The value of 'n' must satisfy the relation as follows –

$$h_{FE}.I_B \geq I_{C\,(sat)}$$

The fan-out of the RTL gate is limited by the value of the output voltage when high. As the output is loaded with inputs of other gates, more current is consumed by the load.

***Q.53. Draw and explain DTL circuit. Enlist its advantages and disadvantages.*** ***(R.G.P.V., June 2017)***

***Ans.*** Fig. 4.36 shows the circuit diagram of DTL NAND gate. The input diodes $D_1$, $D_2$ and $D_3$ conduct through the resistor R, if the corresponding input is in the logic-0 state, while corresponding to logic-1 state the diode is non-conducting. Thus, if at least one of the input is logic-0 state, the diode connected to this input conduct and the voltage $V_P$ at point P is one diode drop above the low level voltage at the input. The voltage $V_P$ should be such as to keep transistor Q is cut-off. Hence, the output of transistor Q is $V_{CC}$. If all the three inputs are in logic-1 state, the input diodes are cut-off and as a result current flowing from $V_{CC}$ through R should be sufficient to drive transistor Q in saturation. Therefore, the output of transistor Q is $V_{CE(sat)}$.

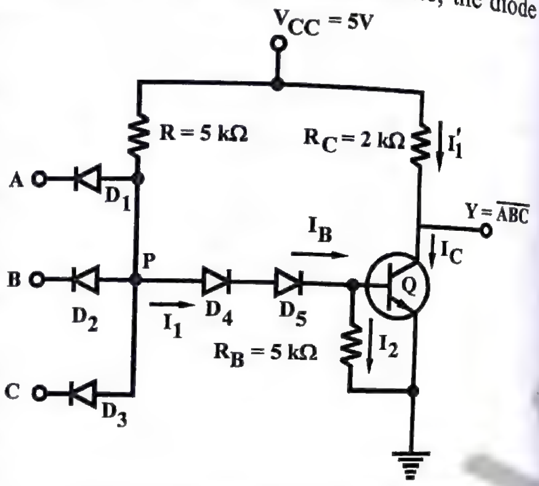


*Fig. 4.36 Basic DTL NAND Gate*

When we consider the voltages corresponding to logic-1 and logic-0 states as $V_{CC}$ and $V_{CE(sat)}$, respectively, this circuit performs NAND operation. The base current ($I_B$) is equal to the difference of currents flowing in the two resistors of 5 kΩ and is sufficient to drive the transistor into saturation. The power dissipation of a DTL NAND gate is about 12 mW and the propagation delay averages 30 n-sec. Noise margin is about 1 V and a fan-out as high as 8 is possible. The fan-out of the DTL NAND gate is limited by the maximum current that can flow in the collector of the saturated transistor.

The various advantages of DTL NAND gate are as follows –

(i) It has wired output capability, because of the resistive load for the output state.

(ii) Compatibility with TTL integrated circuits, which improves flexibility in system design.

(iii) The availability of many functions such that AND, NAND, OR, NOR, EX-OR, etc.

(iv) It has ease of interfacing with discrete circuits.

(v) It requires low power dissipation.

The disadvantages of DTL NAND gate are as follows –

(i) It has limited operating speed.

(ii) Low noise immunity and hence not useful in industrial applications.

(iii) High temperature sensitivity of threshold voltage.

(iv) Only useful for small scale integration (SSI).

(v) It has not ability to drive capacitive loads.

***Q.54. What are logic families ? Explain RTL and DTL.*** ***(R.G.P.V., June 2011)***

***Ans.*** **Logic Families** – Refer the ans. of Q.49.

**RTL and DTL** – Refer the ans. of Q.52 and Q.53.

***Q.55. Draw and explain the TTL logic circuits.*** ***(R.G.P.V., June 2007)***

***Or***

***Write short note on TTL circuits.*** ***(R.G.P.V., June 2008)***

***Ans.*** Fig. 4.37 depicts the TTL NAND gate circuit. The transistor $Q_1$ is multi-emitter transistor, which can be economically fabricated on a chip. Each emitter of transistor $Q_1$ acts like a diode. Therefore, transistor $Q_1$ and resistor $R_{B1}$ act like a three-input AND gate and the remaining circuit inverts the signal. Hence, the overall circuit acts like a three-input NAND gate. To analyze this circuit input, we assume that the load gates are not present and the voltage for logic-0 and logic-1 are $V_{CE(sat)}$ = 0.2 V and $V_{CC}$ = 5V, respectively.

**Working** – When any of the inputs is low (or logic-0), transistor $Q_1$ is 'ON'. The voltage at $B_1$, $V_{B1}$ = (0.2 + 0.7) V = 0.9 V. This voltage is not sufficient to cause transistors $Q_2$ and $Q_3$ to conduct and collector-base junction of transistor $Q_1$ to be forward-biased. The voltage $V_{B1}$ required to be at least (0.7 + 0.5 + 0.5) V = 1.7 V. Thus, transistors $Q_2$ and $Q_3$ are in cut-off state and output is equal to $V_{CC}$ (or 5 V).

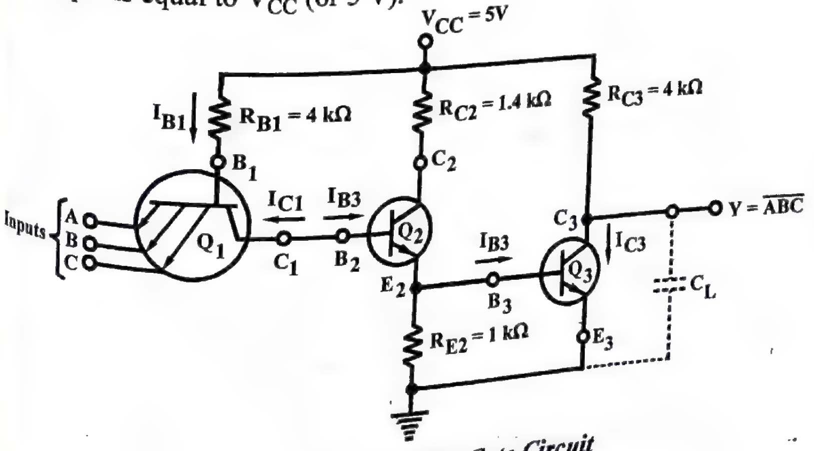


*Fig. 4.37 TTL NAND Gate Circuit*

All inputs are high (or logic-1), the emitter base junctions of transistor $Q_1$ are reverse biased and voltage $V_{B1}$ tends to rise to $V_{CC}$. This cause transistors $Q_2$ and $Q_3$ go into saturation, then $V_{B2} = V_{C1} = (0.8 + 0.8 + 0.7)$ V = 2.3V. The transistor $Q_1$ is operating in the active inverse mode, making current $I_{C1}$ flow in the reverse direction. This current flows into the base of transistor $Q_2$ driving transistors $Q_2$ and $Q_3$ into saturation. So the output is low (or 0.2 V).

When one of the inputs suddenly goes low. The emitter-base junction of transistor $Q_1$ starts conducting and voltage $V_{B1}$drops to 0.9 V. The transistors $Q_2$ and $Q_3$ will be turned-off when the stored base charge is removed. Since $V_{C1} = V_{B1} = 2.3$V, therefore the collector-base junction of transistor $Q_1$ is back-biased, making transistor $Q_1$ operate in the normal active region. This large collector current of transistor $Q_1$ is in direction, which helps in the removal of stored base charge in transistors $Q_2$ and $Q_3$ and improves the speed of circuit. The speed of the circuit can be improved by decreasing $R_{C3}$, which decreases the constant with which the output capacitance charge from low to high states.

***Q.56. Explain the totem-pole output stage of TTL. (R.G.P.V., Dec. 2006)***

***Ans.*** The most commonly used saturating logic family called the transistor-transistor logic (TTL), has the fastest switching speed when compared to other logic families that utilize saturated transistors. This logic family uses a multiple-emitter transistor which is easily and economically fabricated using IC (integrated circuit) techniques.

Fig. 4.38 shows the totem-pole output circuit. In this circuit, the three output components $Q_3$, $Q_4$ and diode $D_1$ are stacked one on the top of the other in the form of totem-pole. At any time, only one of them will be conducting.

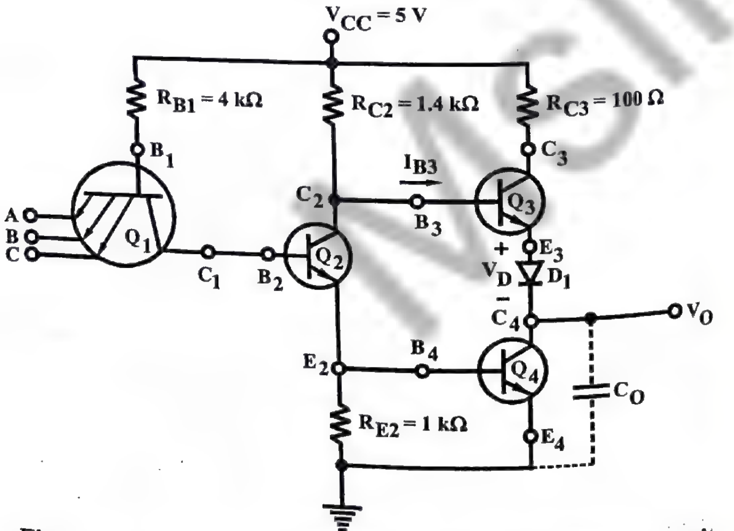


***Fig. 4.38 A TTL NAND Gate with Totem-pole Output Circuit***

**Working** – When all the inputs are high (or logic-1), the emitter-base junction of transistor $Q_1$ is reverse-biased, so that it has no base current. Its collector-base junction is forward-biased and supplying base current $I_{B2}$ to transistor $Q_2$. This current will be sufficiently large to saturate transistor $Q_2$. As a result, transistor $Q_2$ is turned 'ON' and the voltage drop across emitter resistor $R_{E2}$ is sufficient to forward-bias the base-emitter junction of transistor $Q_4$, thereby turning transistor $Q_4$ 'ON' and supplies current for the charging of the output capacitor with a small time constant. This current decreases and eventually becomes zero under steady state condition.

The diode $D_1$ is used in this circuit to keep transistor $Q_3$ in cut-off when the output is at logic-0. Corresponding to this, transistors $Q_2$ and $Q_3$ are in saturation. The voltage drop across the diode keeps the base-emitter junction of transistor $Q_3$ reverse-biased. In this manner, transistor $Q_4$ only conducts when the output is low (or logic-0).

If one of the inputs drops to low (or logic-0), transistors $Q_2$ and $Q_4$ go to cut-off. The output voltage cannot change instantaneously and because of transistor $Q_2$ going to cut-off, the voltage at the base of transistor $Q_3$ rises driving it to saturation.

As the output increases, the base and collector currents of transistor $Q_3$ is decreased and eventually transistor $Q_3$ just comes out of conduction at steady state.

Now, if the output is at V(1) and all the inputs go to high (or logic-1), transistor $Q_2$ goes 'ON'. As a result transistor $Q_3$ and diode $D_1$ go to 'OFF' and transistor $Q_4$ conducts. The output capacitor ($C_O$) discharges through transistor $Q_4$ and as output approaches V(0), transistor $Q_4$ enters into saturation. The output voltage V(1) is obtained as –

$$V(1) = V_{CC} - V_{\gamma(Q_3)} - V_{\gamma(\text{diode})}$$

where, $V_{\gamma(Q_3)}$ = Cut-in voltage of transistor $Q_3$

$V_{\gamma(\text{diode})}$ = Cut-in voltage of diode $D_1$.

***Q.57. What are the advantages and disadvantages of totem-pole output circuit ?***

***Ans.*** Following advantages of totem-pole output circuit are –

(i) The multiple emitter transistor replaces combinations of diodes, resistors and transistors found in other logic circuits.

(ii) Its geometric size is small.

(iii) Smaller size gives lower costs.

(iv) It has high operating speed and high fan-out.

(v) There is no current through resistor $R_{E3}$ in the output low state. Therefore, the inclusion of transistor $Q_3$ and diode $D_1$ keeps the circuit power dissipation low.

(vi) In the high output state, transistor $Q_3$ acts as an emitter follower with its associated low output impedance. This impedance gives a small time constant for charging up any capacitive load on the output.

The disadvantages of totem-pole output circuit are given below –

(i) During the transition of the output from low to high states, transistor $Q_4$ turns-off more slowly as compared to transistor $Q_3$ turns-on and hence there is a period of few nanoseconds during which both transistors $Q_3$ and $Q_4$ are conducting.

(ii) TTL circuit suffers from internally generated current transients or current spikes because of totem-pole connection.

(iii) Totem-pole outputs cannot be wire ANDed, i.e. the outputs of a number of gates cannot be tied together to obtain AND operation of those outputs.

***Q.58. What are the advantages of TTL circuit ? Draw a two input TTL-NAND gate and explain its operation.*** ***(R.G.P.V., Nov. 2018)***

***Ans.*** The advantages of TTL circuit are as follows –

(i) Low manufacturing cost

(ii) Good speed

(iii) Wide range of circuits

(iv) It is the fastest switching speed.

Two-input TTL NAND gate is illustrated in fig. 4.39. In this figure, input transistor $Q_1$ be a multiple emitter transistor and transistor $Q_2$ is known as phase splitter. Diodes $D_1$ and $D_2$ protect $Q_1$ from being damaged by the negative spikes of voltages at the inputs. Diodes conduct and bypass the spikes to ground when the negative spikes appear at the input terminals. Diode D ensures that transistors $Q_3$ and $Q_4$ do not conduct simultaneously. Transistor $Q_3$ works as an emitter follower. Both the base-emitter junctions of $Q_1$ are reverse biased if both the input A and B are high (logic 1 = + 5 V). Then no current flows to the emitters of transistor $Q_1$. Although, the collector-base junction of transistor $Q_1$ is forward biased. So, a current flows via $R_1$ to the base of transistor $Q_2$ and transistor $Q_2$ is turned on. Current from $Q_2$'s emitter flows into the base of transistor $Q_4$. Then transistor $Q_4$ turned on. The collector current of transistor $Q_2$ flows via $R_2$ and therefore, generates a drop across it thereby minimizing the voltage at the collector of transistor $Q_2$. Thus, transistor $Q_3$ is OFF. $V_o$ is at its low level because transistor $Q_4$ is ON. Therefore, the output is zero (i.e. logic 0). The corresponding base-emitter junction is forward biased and the collector-base junction of transistor $Q_1$ is reverse biased when either A or B or both are low. Thus, the current flows to ground via the emitters of transistor $Q_1$. Hence the base of transistor $Q_1$ is at 0.7V, which is not forward bias the base-emitter junction of transistor $Q_2$. So, transistor $Q_2$ is switched off. Transistor $Q_4$ does not obtain the desired base drive with transistor $Q_2$ off. Then transistor $Q_4$ is also switched off. Because transistor $Q_2$ is off, $Q_3$ obtains sufficient base drive. Thus, transistor $Q_3$ is switched on. The output voltage $V_o$ is given by

$$V_o = V_{CC} - V_{R_2} - V_{BE_3} - V_{DD} \simeq 3.4 \text{ to } 3.8 \text{ volt}$$

Thus the circuit works as a two-input NAND gate.

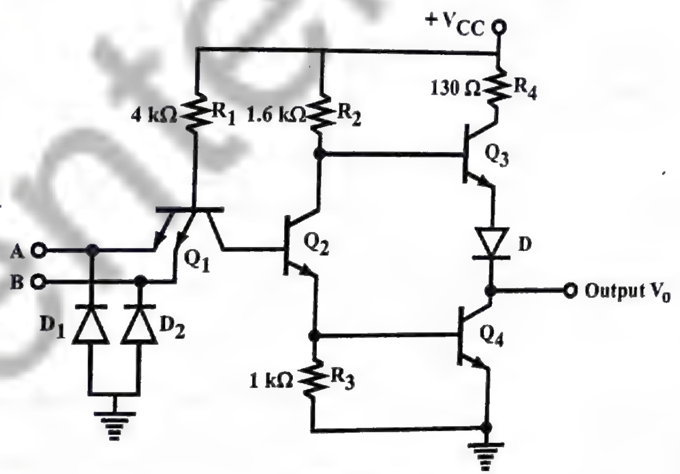


*Fig. 4.39*

***Q.59. Why should totem-pole output not be used for wired AND connection ?***

***Ans.*** The wired-AND connection must not be used for totem-pole output circuits because of the current spike problem. The current spike generates noise in the power supply distribution system and increases power dissipation in the gate. A frequent situation that may arise in wired-AND operation is that some of the gates may be in high (or logic-1) state and other may be in low (or logic-0) state. Under this condition, gates with logic-1 output will source a large amount of current and this will be accepted as sink current by gates in logic-0 state. Thus, the gates in logic-0 state will have to sink a very large sink current, beyond their normal capacity. This will change the logic-0 state output voltage and the output transistor $Q_4$ may be damaged.

***Q.60. Write short note on open-collector TTL.*** ***(R.G.P.V., June 2007)***

***Ans.*** The open-collector TTL gate needs an external resistor that must be connected between the collector of a pull-down transistor and the supply voltage for proper operation. The TTL NAND gate with open-collector output is shown in fig. 4.40. As the collector of tran-sistor $Q_3$ is open, this open collector gate will not work properly unless and external pull-up resistor. The output is taken at the collector of transistor $Q_3$. A high voltage level will appear at the output in the high state.

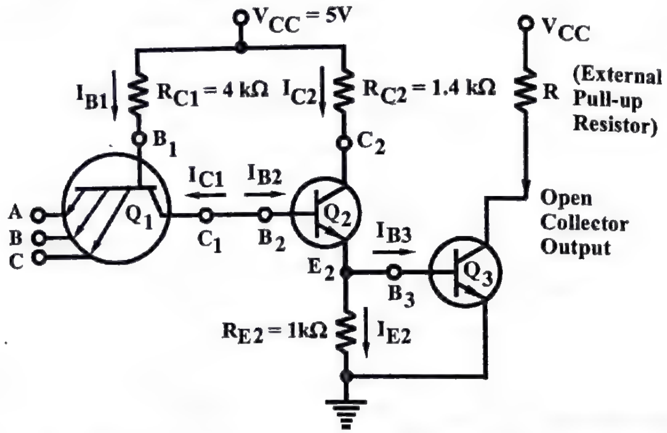


*Fig. 4.40 Open-collector TTL Gate Output with Pull-up Resistor*

The open-collector gates are used, when the wired logic has to be performed as in the simple circuit.

The open-collector gates are used in three main applications, such as driving a lamp or relay, performing wired logic and for the construction of a common-bus system. An open-collector output can drive a lamp placed in its output through a limiting resistor. If the output is low, the saturated transistor $Q_3$ makes a path for the current that turns the lamp 'ON'. When the output transistor is 'OFF', the lamp turns-off because there is no path for the current.

The open-collector arrangement is much slower than the totem-pole arrangement, because the time constant with which the load capacitance charges in this case is considerably larger. The open-collector circuits should not be used in applications where switching speed is a principal importance.

***Q.61. Write short note on ECL logic family.***

***(R.G.P.V., June 2007, Dec. 2010)***

***Or***

***Explain the emitter-coupled logic (ECL) in detail. (R.G.P.V., Dec. 2016)***

***Ans.*** The emitter-coupled logic (ECL) is the fastest of logic families where in high speed operation results from the fact that the transistor operates between active and cut-off regions instead of between saturation and cut-off regions as in other logic families. The ECL operates on the principle of current switching, whereby a fixed bias current less than $I_{C\ (sat)}$ is switched from one transistor's collector to another. Because of this current mode operation, this logic form is also referred to as current-mode logic. It is also called current steering logic (CSL), because current is steered from one device to another.

Fig. 4.41 shows a two-input ECL OR/NOR gate. It has two output which are complements of each other. Transistors $Q_2$ and $Q_{1A}$ form a differential amplifier. Transistor $Q_{1A}$ and $Q_{1B}$ are in parallel. The transistors $Q_3$ and $Q_4$ are emitter followers whose emitter voltages are the same as the base voltages. Inputs are applied to transistors $Q_{1A}$ and $Q_{1B}$. The transistor $Q_2$ is supplied with constant –1.3 V.

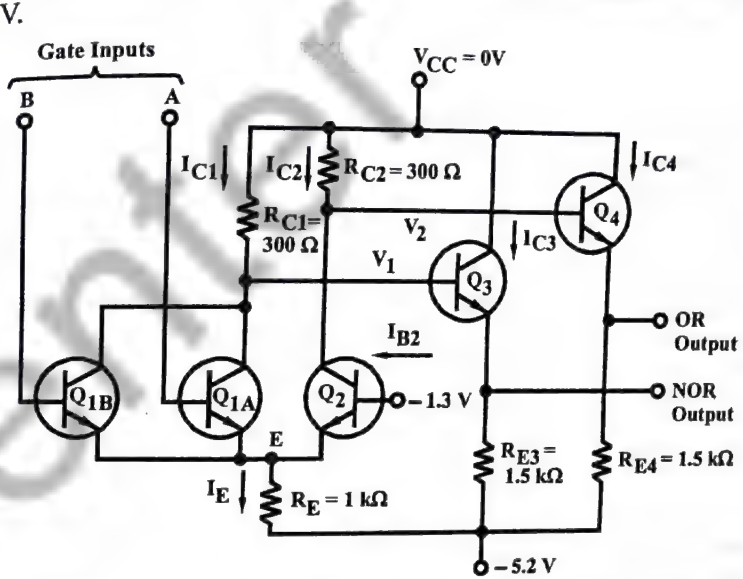


*Fig. 4.41 Circuit Diagram of Two-input ECL OR/NOR Logic*

When both the inputs A and B are low, i.e., –1.7 V, transistor $Q_2$ is more forward biased than transistors $Q_{1A}$ and $Q_{1B}$, therefore transistor $Q_2$ is 'ON' and transistors $Q_{1A}$ and $Q_{1B}$ are 'OFF'. The value of resistor $R_{E2}$ is such that current flowing through transistor $Q_2$ puts the collector at about – 0.9 V. Thus, the emitter of transistor $Q_4$ is at, (– 0.9 – 0.8) V = – 1.7 V and so the OR output is low. The base current of transistor $Q_3$ passing through resistor $R_{E1A}$ is very small. The value of $R_{E1A}$ is such that this current puts the collectors of transistor $Q_{1A}$ and $Q_{1B}$ at about – 0.1 V. Thus, the emitter of transistor $Q_3$ is at, (– 0.1 – 0.8) V = – 0.9 V, i.e., the NOR output is high.

When the input A or B is high, or both A and B are high, the corresponding transistors are 'ON', because they are more forward biased than transistor $Q_2$ and transistor $Q_2$ is 'OFF'. Thus, the collectors of transistor $Q_{1A}$ and $Q_{1B}$ are at – 0.9 V, which makes the NOR output of – 1.7 V, i.e., a logic-0. Only the small base current of transistor $Q_4$ flows through resistor $R_{E2}$. The collector of transistor $Q_2$ is approximately at – 0.1 V and thus the OR output is, – 0.9 V, i.e., at logic-1.

As in ECL logic family the transistors do not go into saturation as is the case with the othei logic families, this ensures faster operation and hence less propagation delay.

***Q.62. What is full form of ECL and $I^2L$ ? Explain ECL in detail.***
***(R.G.P.V., June 2011)***

***Ans.*** Full form of ECL is emitter-coupled logic and $I^2L$ is integrated injection logic.

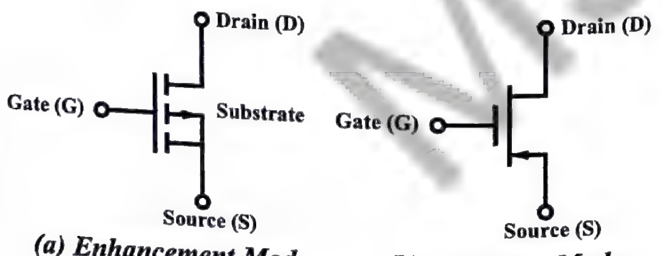
ECL – Refer the ans. of Q.61.

***Q.63. What is MOSFET ?***

***Ans.*** In the Metal Oxide Semiconductor Field Effect Transistor (MOSFET), the channel may be of p-type or n-type, depending on whether the majority carriers are either holes or electrons. The operation mode can be enhancement or depletion, depending on the state of the channel region at zero gate voltage. When the channel is initially doped lightly with p-type impurity, a conducting channel exists at zero gate voltage and the device is said to operate in the depletion mode. If the region beneath the gate is left initially uncharged, a channel must be induced by the gate field before current can flow. Hence, the channel current is enhanced by the gate voltage and such a device is said to operate in enhancement mode.

***Q.64. Write short note on PMOS.***

***Ans.*** In a p-channel MOS, the source terminal is connected to the substrate and a negative voltage is applied to the drain terminal. When the gate voltage is above a threshold ($V_{TH}$) voltage, no current flows in the channel and the drain to source acts as an open circuit. When the gate voltage is sufficiently negative below threshold voltage ($V_{TH}$), a channel is formed and p-type carriers flow from source to drain. The p-type carriers are positive and correspond to a positive current flow from source to drain. The symbol for p-channel MOSFETs is shown in fig. 4.42.



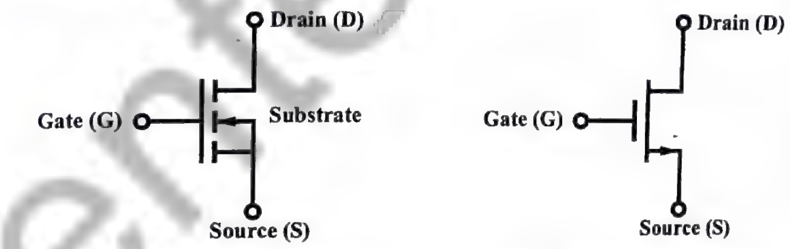
***(a) Enhancement Mode*** ***(b) Depletion Mode***

***Fig. 4.42 Symbol of p-channel MOSFET***

The PMOS uses only p-channel enhancement MOSFET. A resistor at the output of a PMOS circuit could be used to drop the high level voltage to one suitable for CMOS circuit. The holes are current carriers for PMOS.

***Q.65. Write short note on NMOS.***

***Ans.*** In the n-channel MOS, the source terminal is connected to the substrate and a positive voltage is applied to the drain terminal. When the gate voltage is below the threshold voltage ($V_{TH}$), no current flows in the channel. When the gate voltage is sufficiently positive above the threshold voltage ($V_{TH}$) to form the channel, n-type carriers flow from source to drain. The n-type carriers are negative, which corresponds to a positive current flow from drain to source. NMOS uses only n-channel enhancement MOSFET. NMOS has a greater packing density than PMOS, because free electrons are the current carriers in NMOS. The symbol for n-channel MOSFETs is shown in fig. 4.43.

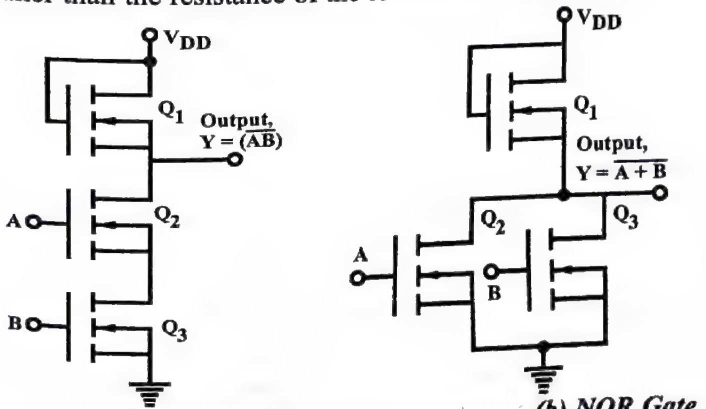


***(a) Enhancement Mode*** ***(b) Depletion Mode***

***Fig. 4.43 Symbol of n-channel MOSFET***

***Q.66. Write short note on NMOS and PMOS logic gates.***
***(R.G.P.V., Nov./Dec. 2007)***

***Ans.*** **NMOS Logic Gate** – Fig. 4.44 (a) shows the circuit of NMOS NAND gate that uses three transistors in series. The inputs A and B must be high for all transistors to conduct and cause the output to go low. When either of the inputs is low, the corresponding transistor is turned-off and the output is high. The series resistance formed by the two active MOS devices must be much smaller than the resistance of the load of MOSFET.

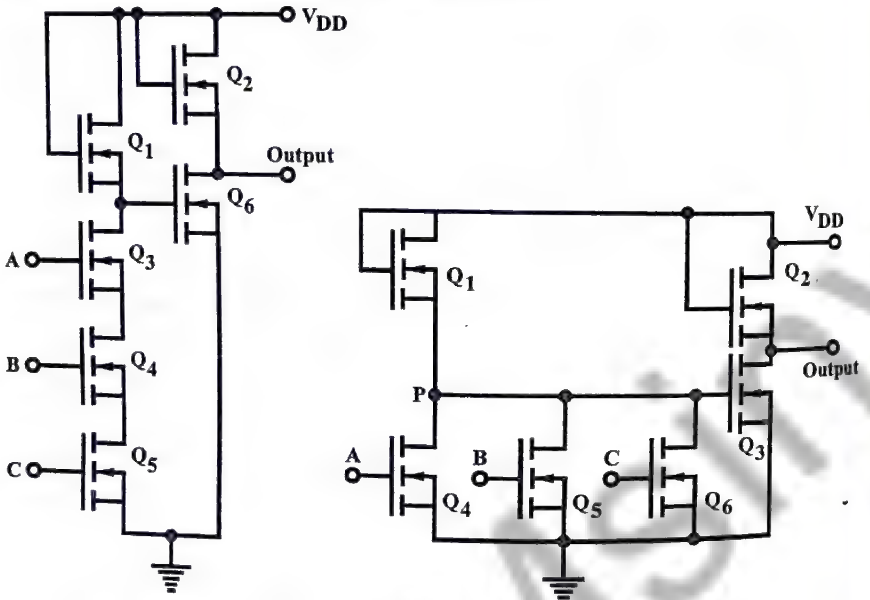


***(a) NAND Gate*** ***(b) NOR Gate***

***Fig. 4.44 n-channel MOS Logic Circuits***

Fig. 4.44 (b) shows the NMOS NOR gate that uses two NMOS transistors in parallel. When either of the input is high, the corresponding MOS transistor conducts and the output is low. When both the inputs are low (or logic-0), both MOSFETs are 'OFF' and thus the output is high (logic-1).

A three-input NMOS AND gate with an additional output inverter is shown in fig. 4.45 (a). The transistors $Q_1$ and $Q_2$ serve as loads. When any one input is at zero volts, then the corresponding input transistor is in 'OFF' state, resulting in conduction of transistor $Q_6$ and this leads zero volt at the output. When all inputs are at a high voltage, the MOSFETs $Q_3$, $Q_4$ and $Q_5$ will be 'ON' and they offer a low resistance compared to the resistance of the load transistor $Q_1$, then the transistor $Q_6$ goes 'OFF'. The output is at a high level. Thus, the circuit performs an AND gate operation.



*(a) AND Gate*

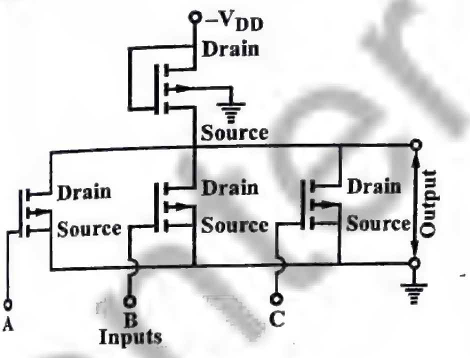
*(b) OR Gate*

***Fig. 4.45 n-channel MOS Logic Circuits***

A three-input n-channel MOS OR gate with an output inverter is depicted in fig. 4.45 (b). The transistors $Q_1$, $Q_4$, $Q_5$ and $Q_6$ form a NOR gate and transistors $Q_2$ and $Q_3$ form an inverter. Transistors $Q_1$ and $Q_2$ act as loads. When the voltage is equal to $V_{DD}$ at one of the inputs, the corresponding transistor is 'ON' and permits the current to flow.

The voltage at point 'P' then drops close to zero volt, the transistor $Q_3$ switches 'OFF' and the output voltage approaches the $V_{DD}$. When all the inputs are low (or logic-0), the transistors $Q_4$, $Q_5$ and $Q_6$ turns-off and the voltage at point 'P' comes near to $V_{DD}$. The transistor $Q_3$ then switches-on and the output drops to zero volt. Thus, this circuit performs the OR operation.

**PMOS Logic Gate –** In fig. 4.46, p-type MOS NAND gate is shown with its truth table. Similarly, we can easily implement other logic gates from PMOS.



| *Inputs* | | | *Output* |
|---|---|---|---|
| A | B | C | D |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

**For positive logic**
**1 = 0 V (turns p-MOS, OFF)**
**0 = –12V (turns p-MOS, ON)**

***Fig. 4.46 PMOS NAND Gate and Truth Table***

The first three MOS transistors act as logic elements and the last simply acts as a load resistor. Suppose $V_{DD}$ = –12V then (referring to positive logic) –

(i) If A, B or C is at –12 V (a logic 0) one or more of the three input transistors will be ON, thereby having a low resistance from drain to source, causing the output to be nearly 0V (a logic 1).

(ii) If A, B and C are all at 0V (a logic 1) then no transistor will conduct and output will approach $V_{DD}$(–12 V) that is, output will be a high (a logic 0). Refer to the truth table fig. 4.46.

***Q.67. Distinguish between PMOS and NMOS logic circuits.***
***[R.G.P.V., June 2002 (EL/ET), 2015]***

***Ans.*** The comparison between PMOS and NMOS with various parameter's are as follows –

| ***Comparison Parameters*** | ***PMOS*** | ***NMOS*** |
|---|---|---|
| Ease of fabrication | (i) The p-channel enhancement MOSFET is easier to fabricate.<br>(ii) The p-channel enhancement MOSFET (or PMOS) is very popular. | (i) The n-channel enhancement MOSFET is not easier to fabricate.<br>(ii) The n-channel enhancement MOSFET (or NMOS) is not popular. |

| | | |
|---|---|---|
| Contamination | In the case of p-channel enhancement MOSFET these contaminant positive ions are being pulled- to the opposite side of the oxide layer to the aluminium $SiO_2$ interface by the negative gate voltage. | In n-channel enhancement MOSFET the gate is normally kept positive w.r.t. the substrate and therefore positive ions collect along the interface between the $SiO_2$ and the substrate. |
| ON Resistance | The p-channel device will have 'ON' resistance more than twice that of n-channel device of the same geometry and operating under the same condition. | The n-channel device have 'ON' resistance less than half that of p-channel device of the same geometry. |
| Switching Speed | The larger junction cross-sectional area of p-channel MOS device and therefore permit lower switching speeds. | The small junction cross-sectional area of n-channel MOS devices and hence permit higher switching speed. |

***Q.68. Write short note on CMOS logic families.***
***(R.G.P.V., June 2012, Dec. 2017)***

***Ans.*** A complementary MOSFET (CMOS FET) is obtained by connecting a p-channel and an n-channel MOSFET in series, with drains tied together and the output is taken at the common drain. Input is applied at the common gate connection formed by connecting the two gates together as shown in fig. 4.47. In a CMOS, p-channel and n-channel enhancement MOS devices are fabricated on the same chip, which makes its fabrication more complicated and reduces the packing density. CMOS is ideally suited for battery operated systems because of negligibly small power consumption.

Its speed is limited by substrate capacitances. To reduce the effect of these substrate capacitances, the latest technology called silicon on sapphire (SOS) is used in microprocessor fabrication, which uses an insulating substrate. The CMOS has become the most popular in medium scale integration and large scale integration areas and is the only possible logic for the fabrication of very large scale integration devices. The CMOS logic gates are used in battery-operated portable equipment.

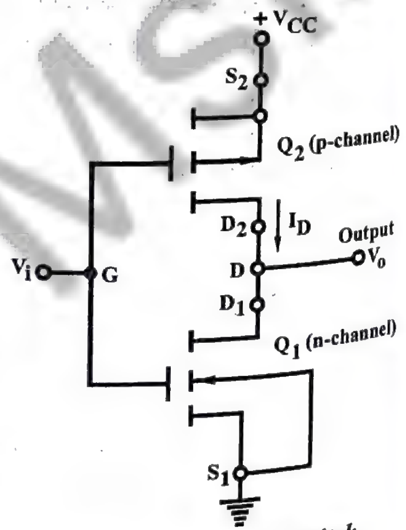


*Fig. 4.47 A CMOS Switch*

***Q.69. CMOS has least power dissipation. Why ?(R.G.P.V., Dec. 2006)***

***Ans.*** In CMOS, there is always a very high resistance between the terminal and ground, because of the MOSFET in current path. Hence its power dissipation is very low.

***Q.70. Explain the working of PMOS, NMOS and CMOS logic.***
***(R.G.P.V., Dec. 2013)***

***Ans.*** Refer the ans. of Q.66 and Q.68.

***Q.71. Describe NMOS and CMOS logic.*** ***(R.G.P.V., June 2011)***

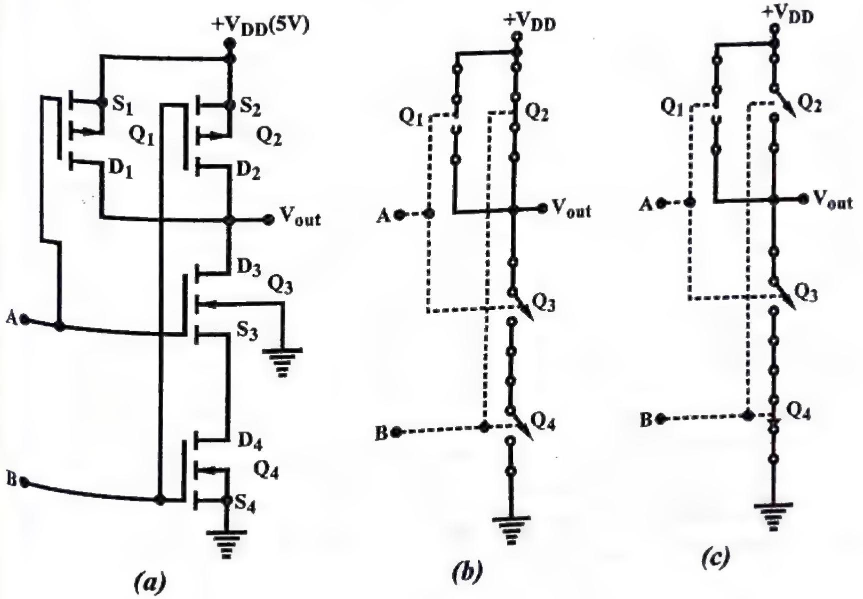
***Ans.*** Refer the ans. of Q.66 and Q.68.

***Q.72. Design a NAND gate using CMOS logic and explain its working.***
***(R.G.P.V., Dec. 2012)***

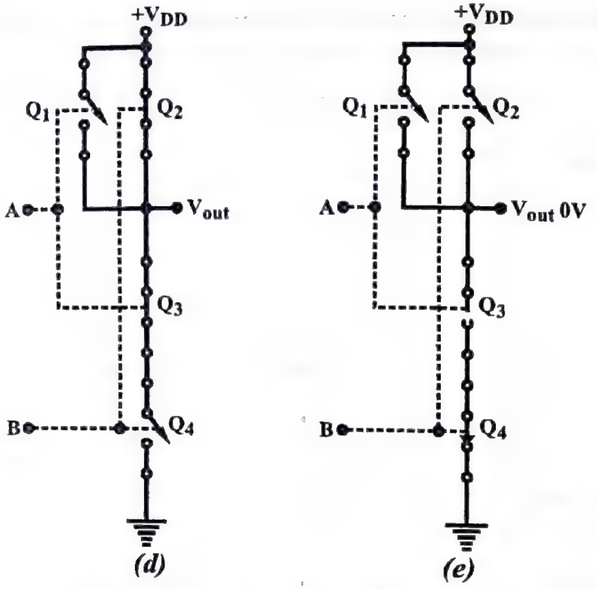
***Or***

***Explain two input CMOS NAND gate with neat diagram.***
***(R.G.P.V., Dec. 2016)***

***Ans.*** A CMOS two-input NAND gate and its equivalent circuits for various input combinations are shown in fig. 4.48. In this figure, $Q_1$ and $Q_2$ are parallel-



*(a)* *(b)* *(c)*



|  | A | B | $Q_1$ | $Q_2$ | $Q_3$ | $Q_4$ | $V_{out}$ |
|---|---|---|---|---|---|---|---|
| (b) | 0 V | 5 V | ON | OF | ON | OFF | 5 V |
| (c) | 0 V | 0 V | ON | OFF | ON | ON | 5 V |
| (d) | 5 V | 5 V | OFF | ON | OFF | OFF | 5 V |
| (e) | 5 V | 0 V | OFF | OFF | OFF | ON | 0 V |

*(f) Truth Table*

***Fig. 4.48 Circuit Diagram and Equivalent Circuits for Various Inputs of the CMOS NAND Gate***

connected PMOS transistors, and $Q_3$ and $Q_4$ are series-connected NMOS transistors. The operation of CMOS NAND gate can be explained as follows –

(i) When A = 0 V and B = 0 V, $V_{GS1} = V_{GS2} = -5$ V, $V_{GS3} = V_{GS4} = 0$ V. Therefore, $Q_1$ is ON, $Q_3$ is OFF, $Q_2$ is ON and $Q_4$ is OFF. Hence, the switching circuit (b) results with $V_{out} = +5$ V.

(ii) When A = 0 V and B = + 5 V, $V_{GS1} = -5$ V, $V_{GS2} = 0$ V, $V_{GS3} = 0$ V, $V_{GS4} = 5$ V. Therefore, $Q_1$ is ON, $Q_3$ is OFF, $Q_2$ is OFF and $Q_4$ is ON. Hence, the switching circuit (c) results with $V_{out} = +5$ V.

(iii) When A = + 5 V and B = 0 V, $V_{GS1} = 0$ V, $V_{GS2} = -5$ V, $V_{GS3} = 5$ V, $V_{GS4} = 0$ V. Therefore, $Q_1$ is OFF, $Q_3$ is ON, $Q_2$ is ON and $Q_4$ is OFF. Hence, the switching circuit (d) results with $V_{out} = +5$ V.

(iv) When A = + 5 V and B = + 5 V, $V_{GS1} = V_{GS2} = 0$ V, $V_{GS3} = V_{GS4} = 5$ V. Therefore, $Q_1$ is OFF, $Q_3$ in ON, $Q_2$ is OFF and $Q_4$ in ON. Hence, the switching circuit (e) results with $V_{out} = 0$ V.

Therefore the circuit is working as a two-input NAND gate. The truth table is depicted in fig. 4.48 (f).

***Q.73. Draw the circuit diagram of 2-input NAND gate (CMOS) and 2-input NOR gate (TTL) and explain their working. (R.G.P.V., Dec. 2015)***

***Ans.*** **2-input NAND Gate (CMOS)** – Refer the ans. of Q.72.

**2-input NOR Gate (TTL)** – Fig. 4.49 shows a circuit diagram of a two-input TTL NOR gate. The transistors $Q_1$ and $Q_2$ are input transistors and $Q_3$ and $Q_4$ are connected in parallel work as a phase splitter as shown in fig. 4.49. The combination of transistors $Q_5$ and $Q_6$ forms an output of totem pole in the circuit.

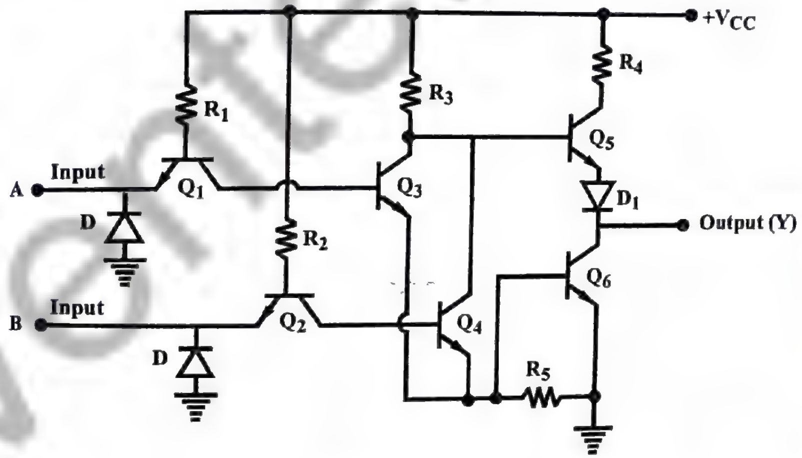


***Fig. 4.49 Two-input TTL NOR Gate***

**Working** – If the inputs are low, the base emitter junctions of transistors $Q_1$ and $Q_2$ are forward-biased and full current away from transistor $Q_1$ and transistor $Q_2$, keeping them OFF. As a result, transistor $Q_5$ is ON and transistor $Q_6$ is OFF, generating a high output.

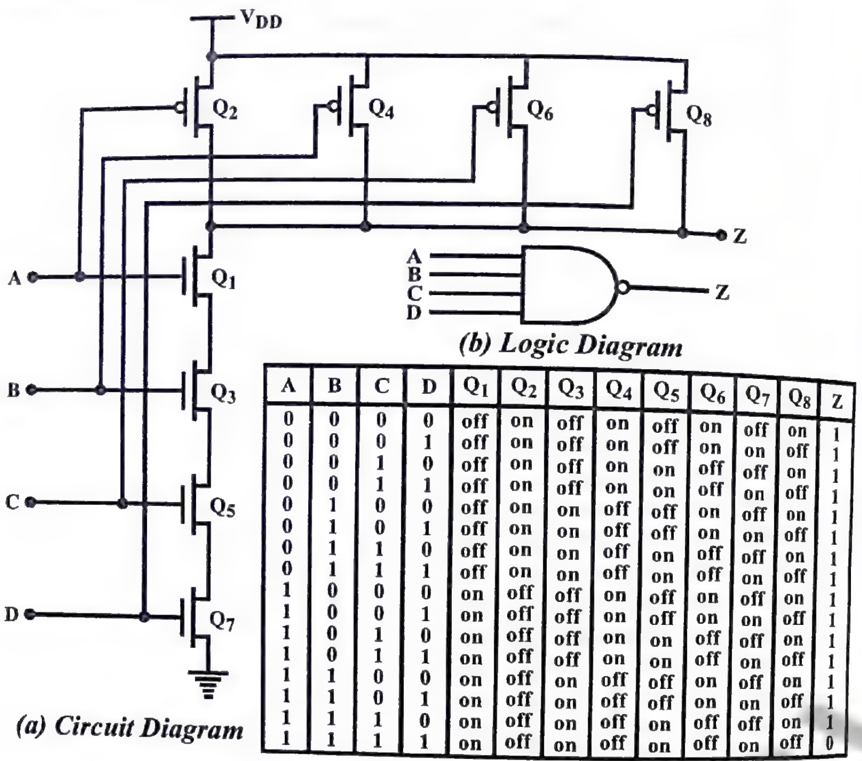
The transistor $Q_3$ is OFF and transistor $Q_4$ is ON when input A is low and input B is high. The transistor $Q_4$ turns OFF $Q_5$ and turns ON $Q_6$, generating a low ouptut.

The transistor $Q_3$ is ON and transistor $Q_4$ is OFF when the input A is high and input B is low. The transistor $Q_3$ turns OFF $Q_5$ and turns ON $Q_6$, generating a low output.

The transistor $Q_3$ and transistor $Q_4$ are ON when both inputs A and B are high. This has the same effects as either one being ON, turning $Q_5$ OFF and $Q_6$ ON. The result is low output.

***Q.74. Show the circuit of four-input NAND gates using CMOS transistors. (R.G.P.V., Dec. 2016)***

***Ans.*** The circuit of four-input NAND gates using CMOS transistors is shown in fig. 4.50.



*(b) Logic Diagram*

| A | B | C | D | $Q_1$ | $Q_2$ | $Q_3$ | $Q_4$ | $Q_5$ | $Q_6$ | $Q_7$ | $Q_8$ | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | off | on | off | on | off | on | off | on | 1 |
| 0 | 0 | 0 | 1 | off | on | off | on | off | on | on | off | 1 |
| 0 | 0 | 1 | 0 | off | on | off | on | on | off | off | on | 1 |
| 0 | 0 | 1 | 1 | off | on | off | on | on | off | on | off | 1 |
| 0 | 1 | 0 | 0 | off | on | on | off | off | on | off | on | 1 |
| 0 | 1 | 0 | 1 | off | on | on | off | off | on | on | off | 1 |
| 0 | 1 | 1 | 0 | off | on | on | off | on | off | off | on | 1 |
| 0 | 1 | 1 | 1 | off | on | on | off | on | off | on | off | 1 |
| 1 | 0 | 0 | 0 | on | off | off | on | off | on | off | on | 1 |
| 1 | 0 | 0 | 1 | on | off | off | on | off | on | on | off | 1 |
| 1 | 0 | 1 | 0 | on | off | off | on | on | off | off | on | 1 |
| 1 | 0 | 1 | 1 | on | off | off | on | on | off | on | off | 1 |
| 1 | 1 | 0 | 0 | on | off | on | off | off | on | off | on | 1 |
| 1 | 1 | 0 | 1 | on | off | on | off | off | on | on | off | 1 |
| 1 | 1 | 1 | 0 | on | off | on | off | on | off | off | on | 1 |
| 1 | 1 | 1 | 1 | on | off | on | off | on | off | on | off | 0 |

*(a) Circuit Diagram*

*(c) Function Table*

*Fig. 4.50 CMOS 4-input NAND Gate*

***Q.75. Compare various logic families in terms of basic gate, power supply, fan-out, power dissipation (mW), propagation delay and noise immunity.*** ***(R.G.P.V., Nov./Dec. 2007)***

***Or***

***Compare the following digital logic families RTL, DTL, TTL, ECL and CMOS.*** ***(R.G.P.V., Dec. 2015)***

***Ans.*** The comparison on the basis of characteristics of all the IC logic families are as follows –

| *Characteristics* | *RTL* | *DTL* | *TTL* | *ECL* | *MOS* | *CMOS* |
|---|---|---|---|---|---|---|
| Basic gate | NOR | NAND | NAND | OR-NOR | NAND | NOR, NAND |
| Power supply (V) | 2.5 | 4.7 | 3.8 | 3.6 | 3.8 | 5 |
| Fan-out | 5 | 8 | 10 | 25 | 20 | > 50 |
| Power dissipation (mW) | 12 | 8-12 | 12-22 | 40-55 | 0.2-10 | 0.01 static at 1 MHz |
| Noise immunity (Noise margin) | Nominal | Good | Very good | Good | Nominal | Very good |
| Propagation delay (ns) | 12 | 30 | 12-6 | 4-1 | 300 | 70 |
| Clock rate ( MHz) | 8 | 12-30 | 15-60 | 60-400 | 2 | 5 |
| Number of functions | High | Fairly high | Very high | High | Low | Low |

In the design of medium speed logic systems, the CMOS logic family is one of the most popular, because it requires low power, it is capable of operating wide range of supply voltage and they can be fabricated into high packing density for a given chip area. The high density has made possible complex functions in a simple chip device.

***Q.76. Give a comparison of the following logic families – DTL, RTL and TTL.*** ***(R.G.P.V., June 2010)***

***Or***

***Compare RTL, DTL and TTL logic families.*** ***(R.G.P.V., Dec. 2017)***

***Or***

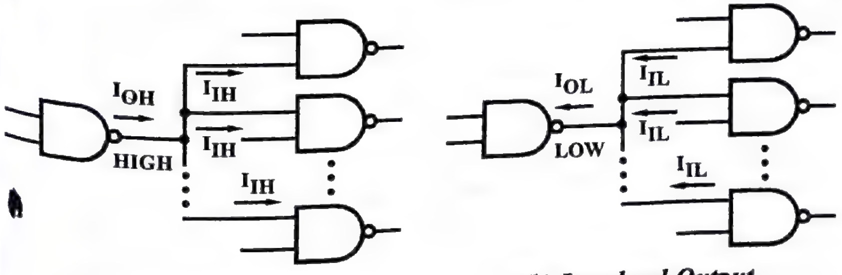
***Compare RTL and DTL logic families.*** ***(R.G.P.V., Dec. 2014)***

***Ans.*** Refer the ans. of Q.75.

***Q.77. Define fan-out and figure of merit. Compare TTL, ECL and CMOS logic families.*** ***(R.G.P.V., May 2018)***

***Ans.* Fan-out** – The fan-out of the logic gate is the maximum number of similar logic gates that a gate can drive without any degradation in voltage levels.

The fan-out is calculated from the amount of current available in the output of a gate and the amount of current required in each input of a gate. Consider the



*(a) High-level Output* *(b) Low-level Output*

*Fig. 4.51 Fan-out Calculation*

circuit connections as shown in figs. 4.51 (a) and (b), respectively. The output of one gate is connected to one or more inputs of other gates. The output of the gate is in the high voltage level in fig. 4.51 (a). It gives a current source of $I_{OH}$ to all the gate inputs connected to it. Each gate input needs a current of $I_{IH}$ for proper operation. Similarly, the output of the gate is in the low voltage level in fig. 4.51 (b). It provides a current sink $I_{OL}$ for all the gate inputs. The fan-out of a gate affects the propagation delay time as well as saturation.

The high state fan-out is,

$$= \frac{I_{OH(max)}}{I_{IH}}$$

where, $I_{OH(max)}$ = Maximum current that the driver gate can source when it is in high state

$I_{IH}$ = Current drawn by each driven gate from the driver gate.

The low state fan-out is,

$$= \frac{I_{OL(max)}}{I_{IL}}$$

where, $I_{OL\ (max)}$= Maximum current that the driver gate can sink when its output is a low state

$I_{IL}$= Current drawn from each driven gate by the driver gate.

**Figure of Merit** – The figure of merit of a digital IC is defined as the product of speed and power. The speed is specified in terms of propagation delay time expressed in nanoseconds.

$$\text{Figure of merit } = \begin{bmatrix}\text{Propagation delay}\\ \text{time in n-sec}\end{bmatrix} \times [\text{Power in mW}]$$

$$= (\text{n-sec}) \times (\text{mW}) = \text{pJ (Pico-joules)}$$

The low value of speed power product is desirable. In a digital circuit, when it is desired to have high speed, then there is a corresponding increase in the power dissipation.

**Comparison** – Refer the ans. of Q.75.

***Q.78. Why interfacing is required ?***

***Ans.*** The output of a circuit or a system should match the input of another circuit or system that has different electrical characteristics. This is referred to as compatibility. Interfacing between different logic families is important for compatibility. An interface circuit is one that is connected between the driver and the load, its function is to take the driver output signal and condition it so that it is compatible with the requirements of the load.

***Q.79. Write short note on interfacing between TTL to MOS.***
***(R.G.P.V., June 2009)***

***Or***

***Discuss about the interfacing between TTL to MOS.***
***(R.G.P.V., June 2012, Dec. 2014)***

***Or***

***How is interfacing TTL to MOS obtained.*** ***(R.G.P.V., Dec. 2012)***

***Or***

***How a TTL gate can drive N-CMOS gates ? Explain with example.***
***(R.G.P.V., June 2014)***

***Ans.*** **TTL to CMOS Interfacing** – The MOS and CMOS gates are slower than the TTL gates, but consume less space. Thus, there is an advantage in using TTL and MOS devices in combination. The input current values of CMOS are extremely low than the output current capabilities of any TTL series. Thus, TTL has no problem in meeting the CMOS input current requirements. A level translator is used to raise the level of the output voltage of the TTL gate to an acceptable level for CMOS, as shown in fig. 4.52 (a), where the TTL output is connected to a + 5V source with a pull-up resistor. The pull-up resistor will cause the TTL output to rise to the +5V approximately in the high state, thereby providing an adequate CMOS input.

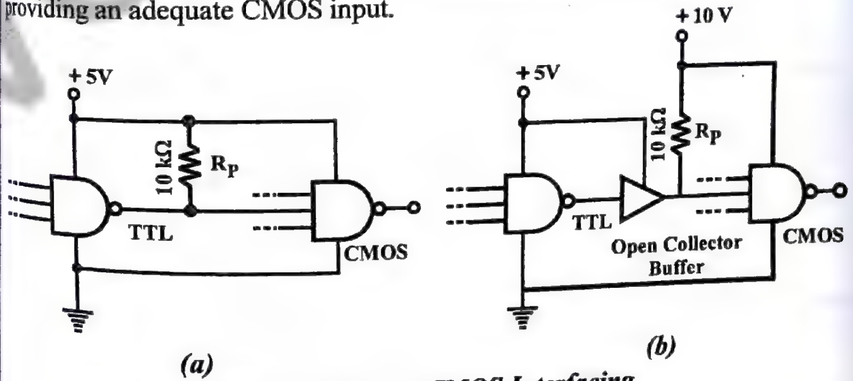


***Fig. 4.52 TTL to CMOS Interfacing***

When the TTL has to drive a high voltage CMOS, the pull-up resistor cannot be used to raise the level of the TTL output to the level of the CMOS input, since the TTL is sensitive to voltage levels.

In such a case, an open collector buffer can be used to interface TTL to a high voltage CMOS as illustrated in fig. 4.52 (b).

Fig. 4.53 illustrates a TTL gate driving n-CMOS gates. The following conditions are needed to be satisfied for such circuit to operate properly.

$V_{OH}$ (TTL) ≥ $V_{IH}$ (CMOS) ...(i)

$V_{OL}$ (TTL) ≤ $V_{IL}$ (CMOS) ...(ii)

$-\ I_{OH}$ (TTL) ≥ n $I_{IH}$ (CMOS) ...(iii)

$I_{OL}$ (TTL) ≥ – n $I_{IL}$ (CMOS) ...(iv)

All the conditions illustrated in equations (i), (ii), (iii) and (iv) are always satisfied in case of 74HCT and 74ACT series for high value of n. This shows that these two CMOS series are TTL compatible.

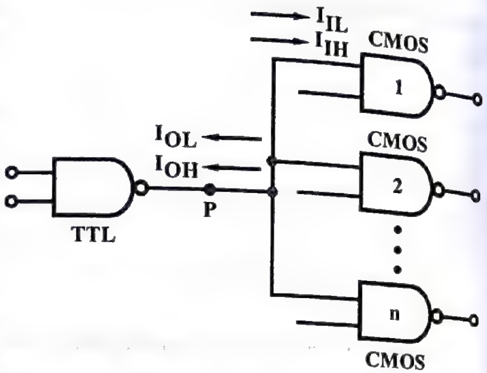


*Fig. 4.53 A TTL Gate Driving n-CMOS Gates*

**CMOS to TTL Interfacing** – The CMOS output can supply enough voltage and current to satisfy the TTL input requirements in the high state. Thus, no special consideration is required for the high state. The TTL input current requirements at low state cannot be directly met. Hence, an interface circuit with a low input current requirement and a sufficiently high output current rating is required.

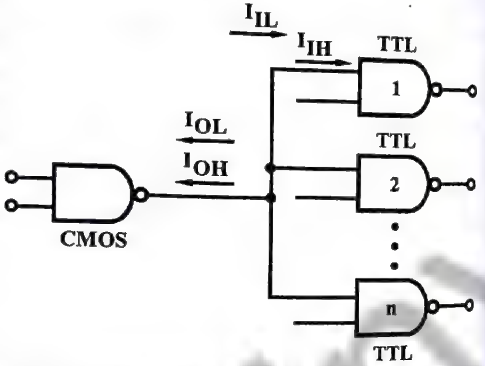


*Fig. 4.54 CMOS Gate Driving n-TTL Gates*

The 74 C series of CMOS ICs can be operated for any supply voltage in the range of 3V to 15V, whereas the 74HC/74HCT/74AC/74ACT series have the supply voltage range of 2V to 6V. Since the supply voltage used for all 74 series, TTL ICs is 5V, therefore, it is necessary to operate CMOS devices at + 5V to make it compatible with TTL gates.

Fig. 4.54 illustrates a CMOS gate driving n-TTL gates. The following conditions are required to be satisfied for such an arrangement to operate properly.

$$V_{OH}\text{ (CMOS)} \geq V_{IH}\text{ (TTL)} \quad ...(v)$$

$$V_{OL}\text{ (CMOS)} \leq V_{IL}\text{ (TTL)} \quad ...(vi)$$

$$-I_{OH}\text{ (CMOS)} \geq n\, I_{IH}\text{ (TTL)} \quad ...(vii)$$

$$I_{OL}\text{ (CMOS)} \geq -n\, I_{IL}\text{ (TTL)} \quad ...(viii)$$

The conditions of equations (v) and (vi) are always satisfied. When 74ACT CMOS is driving 74ALS TTL gates, the noise margins as –

$$\Delta\ 1 = (3.76 - 2)\text{ V} = 1.76\text{ V}$$

$$\Delta\ 0 = (0.8 - 0.37)\text{ V} = 0.43\text{ V}$$

The conditions of equations (vii) and (viii) are always satisfied for 74HC/74HCT/74AC/74ACT CMOS series. The value of n is different for different series.

# UNIT 5

## INTRODUCTION TO DIGITAL COMMUNICATION, NYQUIST SAMPLING THEOREM, TIME DIVISION MULTIPLEXING

***Q.1. Give a brief introduction of digital communication.***

***Ans.*** Digital communication involves the transmission of information in digital form (transformed and encoded) from a source via a channel to the destination through a decoder-receiver. Although the past decade has witnessed major advancement in the theory of digital communication, the basic concept of digital communication is quite old. The very first form of electrical communication was digital in nature. Due to the availability of wideband communication channels and tremendous development in the area of VLSI (very large scale integration) technology, the successful development of digital communication has been possible. The wideband communication channels cater the require to increase transmission bandwidth of digital communication systems, whereas the VLSI technology offers a cost effective way of realizing the building hardware of the digital communication system.

***Q.2. What are the advantages of digital technology over analog technology ?***

***Ans.*** (i) Maintenance of digital system is easier than maintenance of an analog one.

(ii) Data rate of digital system is much higher than analog system.

(iii) Digital transmission is much cheaper than analog transmission, since it is not necessary to accurately reproduce an analog waveform after it has passed through potentially hundreds of amplifiers on a trans continental call.

(iv) In digital transmission, voice, data, music and images (television, fax and video) can be interspread to make more efficient use of the circuits and equipment.

(v) In a digital technology digital signal can pass through an arbitrary number of regenerators with no loss in signal and thus travel long distances with no information loss.

(vi) It is easy to calculate how far a signal can propagate and still be recognizable.

***Q.3. Discuss the elements of digital communication system with the help of block diagram.***

***Ans.*** A digital communication system arrangement is illustrated in fig. 5.1. The message input source is converted to electrical analog form using an input transducer. The information or message to be conveyed can be available in either analog or digital form in digital communication. Then the source output is converted into a binary digits sequence to efficient representation and transmission over a specific channel. This is performed by encoder. Source encoding or data compression is defined as the process of showing the data to be sent by an efficient technique. After source encoding, some redundancy is introduced into the encoded data to make appropriate for transmission to the channel. This helps to minimise the noise effect and other interference. This is called channel encoding. At the output of encoder, the binary sequence is passed via a digital modulator. The main function of digital modulation is to map the coded binary sequence into signal waveform. This is required for transmission because practical channels support electrical signals. Then the modulated signal is launched into channel. The channel is a physical medium that communicates the sender and receiver. In the channel, the sent signal gets corrupted by a several unwanted signal known as noise increasing from different source.

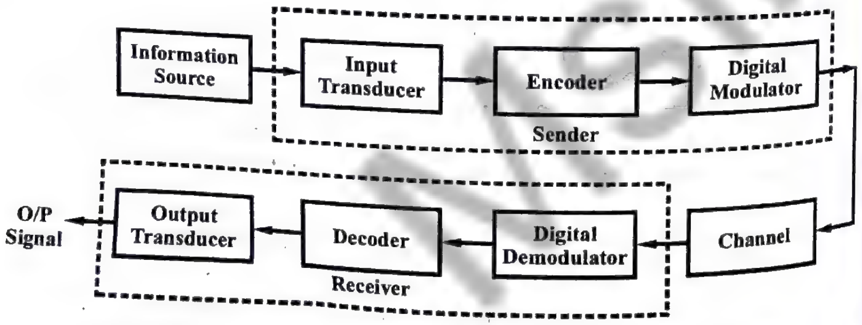


***Fig. 5.1 Block Diagram of Digital Communication Systems***

At the receiver, the main function of digital demodulator is to convert the received modulated signal corrupted through channel noise to a binary coded data sequence which was originally sent. Then binary data sequence is passed via decoder. The signal regenerated by the decoder is an approximation of the original information at the source output because of noise and other disturbances. The purpose of output transducer is to convert the reconstructed electrical signal into the original signal form of information.

***Q.4. State and explain sampling theorem.***

***Or***

***Write short note on Nyquist sampling theorem. (R.G.P.V., Nov. 2018)***

***Ans.*** Consider a signal m(t) which has the highest-frequency spectral component $f_M$, the sampling frequency $f_s$ must be no less than $f_s = 2f_M$ only when the lowest-frequency spectral component of m(t) is $f_L = 0$. In the normal case, where $f_L \neq 0$, it may be that the sampling frequency required be no larger than $f_s = 2(f_M - f_L)$. For instance, the signal may be recovered from samples taken at a frequency $f_s = 2\ (10.1 - 10.0) = 0.2$ MHz, if the spectral range of a signal varies from 10.0 to 10.1 MHz.

To explain the theorem of sampling for these type of bandpass signals, let us choose a sampling frequency $f_s = 2\ (f_M - f_L)$ and let us consider at beginning that it happens that the frequency $f_L$ turns out to be an integral multiple of $f_s$, i.e., $f_L = nf_s$ with n an integer. This situation is shown in fig. 5.2. The double-sided spectral pattern of a signal m(t) with Fourier transform M(jω) is shown in fig. 5.2 (a). Here it has been arranged that n = 2; is, $f_L$ coincides with the second harmonic of the sampling frequency, while the sampling frequency is exactly $f_s = 2\ (f_M - f_L)$. The spectral pattern of the sampled signal S(t)m(t) is shown in fig. 5.2 (b). The dc term of S(t) and the product of m(t) duplicates in part (b) the form of the spectral pattern in part (a) and leaves it in the similar frequency range from $f_L$ to $f_M$. The product of m(t) and the spectral component in S(t) of frequency $f_s$ (= $l/T_s$) gives rise in part (b) to a spectral pattern derived from part (a) by shifting the pattern in part (a) to the right and also to the left by amount $f_s$. Similar to this, the higher harmonics of $f_s$ in S(t) give rise to corresponding shifts, right and left, of the spectral pattern in part (a). Now we noted that when the sampled signal S(t)m(t) is passed through a bandpass filter with arbitrarily sharp cutoffs and with passband from $f_L$ to $f_M$, the signal m(t) will be recovered exactly.

The m(t) spectrum extends over the first half of the frequency interval between harmonics of the sampling frequency, i.e., from $2.0f_s$ to $2.5f_s$ as shown in fig. 5.2. As a outcome, there is no spectrum overlap, and signal recovery is possible. When m(t) spectral range extended over the second half of the interval from 2.5 $f_s$ to 3.0 $f_s$, there would also be no overlap. Let, however, that the m(t) spectrum were confined neither to the first half nor to the second half of the interval between sampling-frequency harmonics. In this type of case, there would be overlap between the patterns of spectrum, and signal recovery would not be possible. Thus the minimum sampling frequency allowable is $f_s = 2(f_M - f_L)$ given that either $f_M$ or $f_L$ is a harmonic of $f_s$. Fig. 5.3 (a) shows the spectrum of the bandpass signal and fig. 5.3 (b) shows the spectrum of NS shifted by the $(N-1)^{st}$ and the $N^{th}$ harmonic of the sampling waveform.

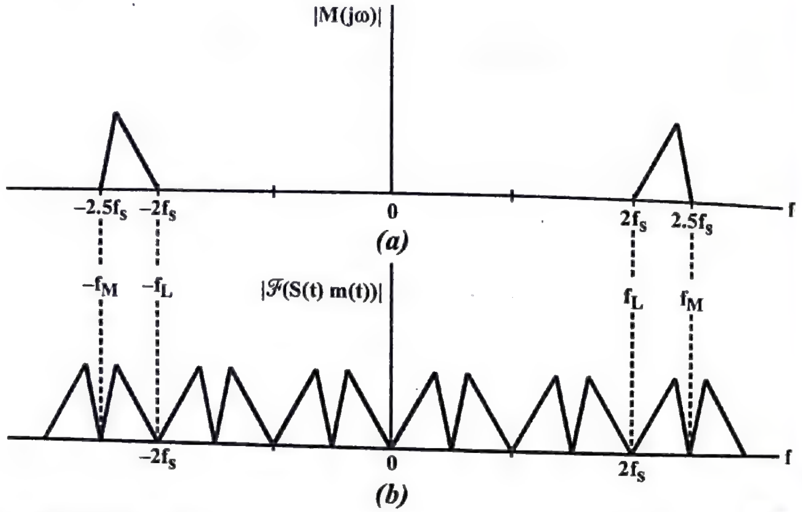


***Fig. 5.2 (a) The Spectrum of a Bandpass Signal (b) The Spectrum of the Sampled Bandpass Signal***

A simple analysis is needed when $f_L$ and $f_M$ are not a harmonic of $f_s$. Regenerated spectral pattern of fig. 5.2 is shown in fig. 5.3 (a). PS and NS are the positive and negative frequency part of the spectrum, respectively. The manner in which NS and PS are shifted because of sampling and NS and PS are considered separately for simplicity and assume initially what constraints should be imposed so that we cause no overlap over, say, PS.

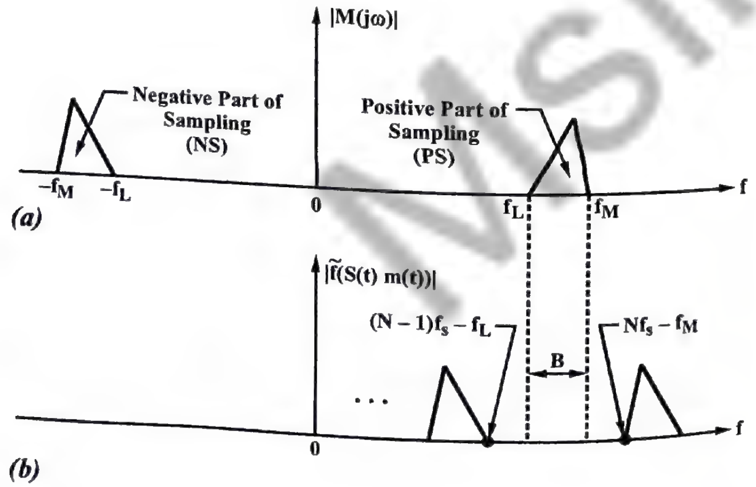


***Fig. 5.3***

The right shifted patterns of NS are shown in fig. 5.3 (b) because of the $(N-1)^{st}$ and $N^{th}$ harmonics of the sampling waveform. To eliminate overlap it is essential that

$$(N-1)f_s - f_L \leq f_L \qquad \text{...(i)}$$

and
$$Nf_s - f_M \geq f_M \qquad \text{...(ii)}$$

so that, with $B \equiv f_M - f_L$ we have

$$(N-1)f_s \leq 2(f_M - B) \qquad \text{...(iii)}$$

and
$$Nf_s \geq 2f_M \qquad \text{...(iv)}$$

If we let $k \equiv f_M/B$, equations (iii) and (iv) become

$$f_s \leq 2B\left(\frac{k-1}{N-1}\right) \qquad \text{...(v)}$$

and
$$f_s \geq 2B\left(\frac{k}{N}\right) \qquad \text{...(vi)}$$

Where $k \geq N$ since $f_s \geq 2B$. Equations (v) and (vi) establish the constraints which must be observed to eliminate an overlap on positive spectrum. It is noted from the symmetry of the initial spectrum and the symmetry of the shiftings needed that this same constraints assures that there will be no overlap on negative spectrum.

***Q.5. What is sampling ? Explain various types of sampling.***

***Ans.*** Basically, sampling refers to uniform sampling which converts an analog signal into a sequence of samples that are uniformly spaced in time. Sampling process is necessary to select the sampling raie suitably so that the sequence of samples can uniquely define the original analog signal.

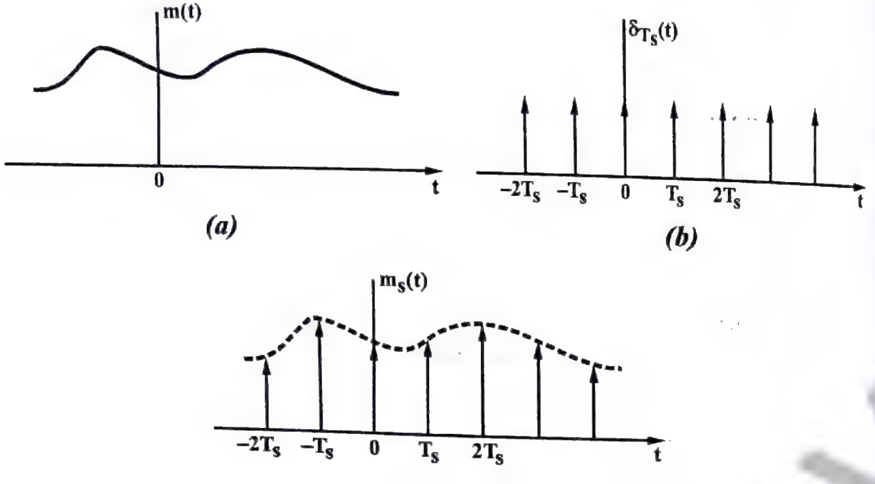
To generate sampled signal, we use generally following types of sampling –

***(i) Instantaneous Sampling*** – We get an infinite sequence of samples $\{m(nT_s)\}$ if we sample an arbitrary signal m(t) at a uniform rate and instantaneously, once each $T_s$ second, where n takes on all possible integer values. This ideal form of sampling is called instantaneous sampling. This is shown in fig. 5.4 (a).

***(ii) Ideal Sampled Signal*** – We get the following on multiplying m(t) by a unit impulse train $\delta_{T_s}(t)$ [see fig. 5.4 (b)] –

$$m(t)\,\delta_{T_s}(t) = m_s(t) = \sum_{n=-\infty}^{\infty} m(nT_s)\,\delta(t-nT_s)$$

Ideal sampled signal is represented by $m_s(t)$ [refer fig. 5.4 (c)].

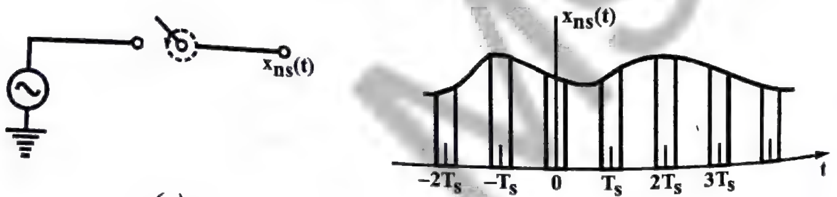


*(a)* *(b)*

*(c)*

***Fig. 5.4 Ideal Signal Sampling***

***(iii) Natural Sampling*** – In practice, sampling of band-limited analog signal m(t) is performed by high speed switching circuits. Fig. 5.5 (a) and (b) show an equivalent circuit employing a mechanical switch and resulting sampled signal, respectively.



*(a)* *(b)*

***Fig. 5.5 Natural Sampling***

The sampled signal,

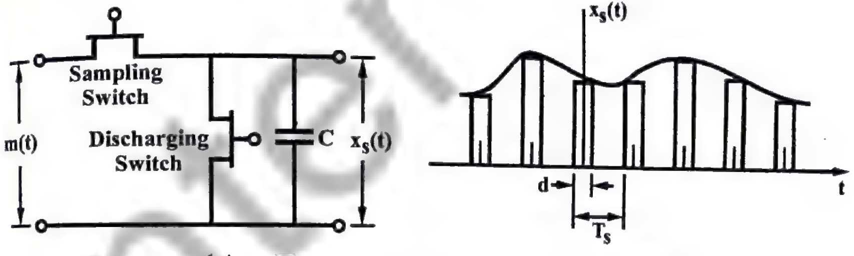
$$x_{ns}(t) = m(t)\ x_p(t)$$

where $x_p(t)$ = Periodic train of rectangular pulses with periods $T_s$.

In $x_p(t)$ every rectangular pulse has width d and unit amplitude.

***(iv) Flat-top Sampling*** – A functional block termed sample and hold circuit performes a very popular practical sampling method as shown in fig. 5.6 (a).

A flat-top sampled signal $x_s(t)$ produced by this circuit is shown in fig. 5.6 (b).



*(a)* *(b)*

***Fig. 5.6 Flat-top Sampling***

***Q.6. Write the differences between the ideal sampling and the reconstruction techniques. Also explain aperture effect.***

***Ans.*** There are a number of differences between the ideal sampling and reconstruction techniques are given below –

(i) The sampled wave in practical systems consists of finite amplitude and finite duration pulses rather than impulses.

(ii) The waveforms that are sampled are often timelimited signals and hence are not bandlimited.

(iii) Reconstruction filters in practical systems are not ideal filters.

The effects of these differences on the quality of the reconstructed signals. The sampled waveform produced by practical sampling devices, especially the sample and hold variety has the form,

$$x_s(t) = \sum_{k=-\infty}^{\infty} x(kT_s)\,p(t-kT_s)$$

$$= [p(t)] * \left[\sum_{k=-\infty}^{\infty} x(kT_s)\delta(t-kT_s)\right]$$

where p(t) is a flat-topped pulse of duration τ. The spectrum $X_s(f)$ of $x_s(t)$ is given by

$$X_s(f) = P(f)\ X_\delta(f) = P(f)\left[f_s \sum_{n=-\infty}^{\infty} X(f-nf_s)\right]$$

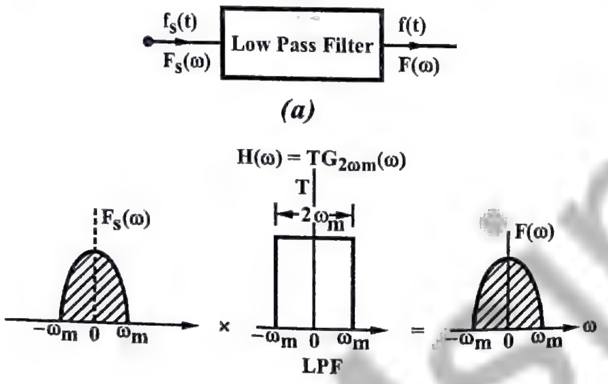
where P(f) is the Fourier transform of p(t) and $X_\delta(f)$ is the Fourier transform of the ideal sampled wave. P(f) is a sine function and hence we can say from above equation that the primary effect of flat topped sampling is an attenuation of high frequency components. This effect, sometimes called an ***aperture effect***, can be compensated by an equalizing filter with a transfer function $H_{eq}(f) = 1/P(f)$. However, if the pulsewidth is chosen to be small compared to the time between samples then P(f) is essentially constant over the message band and no equalization may be needed. Thus effects of pulse shape are often unimportant and flat topped sampling is a good approximation to ideal impulse sampling.

***Q.7. Discuss sampling theorem. Explain the generation of sampled signal and how the original signal is recovered from a sampled signal. Also focus on alising effect.*** ***(R.G.P.V., Dec. 2008)***

*Ans.* **Sampling Theorem** – Refer the ans. of Q.4.

**Generation of Sampled Signal** – Refer the ans. of Q.5.

**Recovery of Original Signal from Sampled Signal** – Fig. 5.7 (a) shows the original signal f(t), recovery from its sampled version $f_s(t)$.



*(a)*

*(b)*

***Fig. 5.7 A Function Recovery from its Sampled Version***

The cut-off frequency of the low-pass filter is $\omega_m$ radians. The transfer function of a low pass filter is a gate function. Thus the baseband spectrum $F(\omega)$ may be recovered from $F_s(\omega)$ by multiplying the latter with a gate function which is shown in fig. 5.7 (b). From the figure we get

$$F_s(\omega)\cdot G_{2\omega m}(\omega) = \frac{1}{T}F(\omega)$$

or

$$F(\omega) = F_s(\omega)\cdot TG_{2\omega m}(\omega) \qquad ...(i)$$

A low pass filter with a cut-off frequency $\omega_m$ is represented by the gate function $G_{2\omega m}(\omega)$. Now, we can say that the action of low-pass filtering is equivalent to multiplying the sampled signal $F_s(\omega)$ with a gate function $TG_{2\omega m}$ and this action yields the baseband spectrum $F(\omega)$.

By using the time convolution theorem, the time domain equivalent of RHS of equation (i) can be evaluated i.e.,

$$f(t) = f_s(t) \otimes T\frac{\omega_m}{\pi}\,Sa(\omega_m t)$$

$$= f_s(t) \otimes T\frac{\omega_0}{2\pi}\,Sa(\omega_m t) \qquad \left[\because \omega_m = \frac{\omega_0}{2}\right]$$

$$= f_s(t) \otimes Sa(\omega_m t) \qquad \left[\because \frac{1}{T} = \frac{\omega_0}{2\pi}\right] \quad ...(ii)$$

Considered that the sampled function $f_s(t)$ is a sum of impulses located at sampling instants nT, having strength equal to sample value $f_n$ at that instant as shown in fig. 5.8 (a). Thus,

$$f_s(t) = \sum_n f_n\,\delta(t - nT) \qquad ...(iii)$$

here $f_n$ is the nth sample of f(t).

Substituting $f_s(t)$ from equation (iii) into equation (ii), we have

$$f(t) = \sum_n f_n\,\delta(t - nT) \otimes Sa(\omega_m t)$$

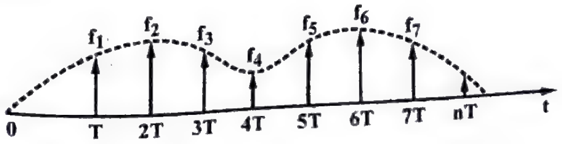
By using sampling property of delta function, we have

$$f(t) = \sum_n f_n\,Sa[\omega_m (t - nT)]$$

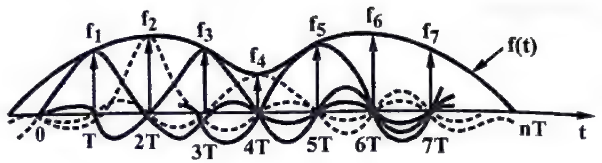
Putting $\omega_m T = \pi$,

$$f(t) = \sum_n f_n\,Sa[\omega_m t - n\pi] \qquad ...(iv)$$

It is clear from equation (iv) that f(t) function may be constructed by multiplying a sampling function $Sa(\omega_m t - n\pi)$ with $(f_n)$, the samples of f(t) and adding the multiplied values. $Sa[\omega_m(t - nT)]$ indicates the sampling functions at sampling instants t = nT. Fig. 5.8 (b) shows the this way of recovery of f(t).



***(a) Sampled Function***



*(b) Reconstruction of f(t) in Time Domain*

*Fig. 5.8*

**Nyquist Interval –** From the equation

$$T \leq \frac{1}{2f_m} s \qquad ...(v)$$

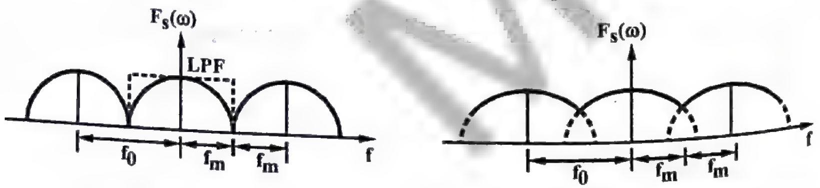
it is clear that maximum sampling interval is described as given below –

$$T = \frac{1}{2f_m} s \qquad ...(vi)$$

Equation (vi) is known as ***Nyquist sampling interval.***

If the bandlimited signal is sampled at Nyquist rate the $F_s(\omega)$ spectrum will have non overlapping $F(\omega)$ repeating periodically, however, every spectrum $F(\omega)$ will be connecting the neighbouring ones as shown in fig. 5.9 (a).

**Aliasing Effect –** As shown in fig. 5.9 (b), if a bandlimited signal is sampled at rate lower than Nyquist rate, $f_0 < 2f_m$ or Nyquist interval $[T > (1/2)f_m]$ is lower than sampling interval then periodically repeating $F(\omega)$ in the spectrum of sampled signal overlap with neighbouring ones. In this case, the signal is under-sampled and some aliasing is generated in this under-sampling method. Aliasing refers to the phenomenon in which an element of high frequency in the signal spectrum apparently taking on the identity of a lower frequency in the spectrum of its sampled version. Due to aliasing phenomenon, it is no longer possible to recover f(t) from $f_s(t)$ by low pass filter since the spectral components in the overlap regions add and so the signal is distorted. Solid curve in fig. 5.9 (b) represents the aliasing spectrum.



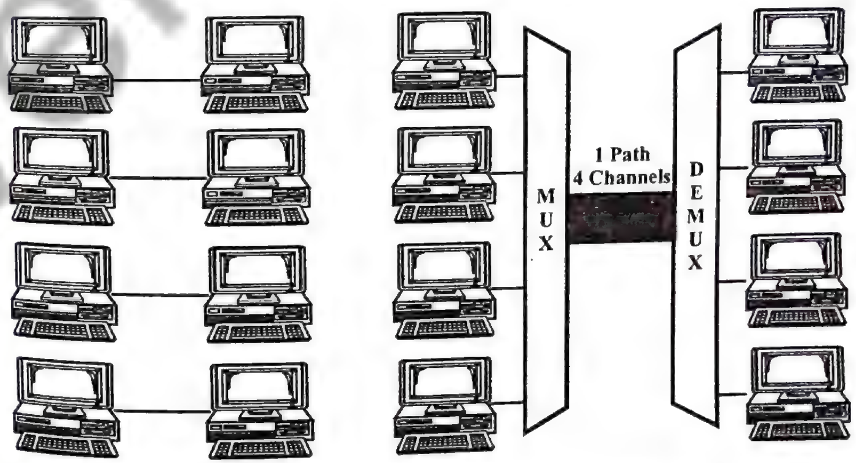
*(a) Nyquist Rate* *(b) Slower than Nyquist Rate*

*Fig. 5.9 Spectrum of f(t) Sampled*

We may use prior to sampling an anti-aliasing low pass filter, in order to combat the aliasing effect the filtered signal is then sampled at a rate slighty higher than the Nyquist rate.

***Q.8. Write short note on multiplexing.***

***Ans.*** Multiplexing is the set of techniques which permit the simultaneous transmission of multiple signals across a single data link. Whenever the transmission capacity of a medium linking two devices is greater than the transmission needs of the device, the link can be shared, much as a large water pipe can carry water to several separate houses at once. As data-and telecommunications usage increases, so does the traffic. This increase can be accommodated by continuing to add individual lines each time a new channel is required or higher capacity links can be installed and each is used to multiple signals. Modern technology includes high-bandwidth media such as coaxial cable, optical fibre, and terrestrial and satellite microwaves. Each of these has a carrying capacity far in excess of that required for the average transmission signal. When the transmission capacity of a link is greater as compared to the transmission requirements of the devices connected to it, the excess capacity is wasted. An efficient system maximizes the utilization of all facilities.



*(a) No Multiplexing* *(b) Multiplexing*

*Fig. 5.10 Multiplexing versus No Multiplexing*

In a multiplexed system, n devices share the capacity of one link. Fig. 5.10 shows two possible ways of linking four pair of devices. In fig. 5.10 (a), each pair has its own link. If the full capacity of each link is not being utilized, a portion of that capacity is being wasted. Fig. 5.10 (b) shows the basic format of a multiplexed system. The four devices on the left direct their transmission streams to a ***multiplexer (MUX)***, which combines them into a single stream (many to one). At the receiving end, that stream is fed into a ***demultiplexer (DMUX)***, which separates the stream back into its component transmissions (one to many) and directs them to their intended receiving devices

***Q.9. Write short note on 'TDM'.***

***Ans.*** The technique of separating the signal in time is called ***time-division multiplexing (TDM).*** The TDM system is shown in fig. 5.11. Each input message signal is first restricted in bandwidth by a low-pass (pre-alias) filter to remove the frequencies, which are non-essential to an adequate signal representation. The low-pass filter output are then applied to a commutator, which is usually implemented using electronic switching circuitry. The function of the commutator is two fold –

(i) To take a narrow sample of each of the N-input messages at a rate of $f_s$ that is slightly higher than 2W, where W is the cut-off frequency of the pre-alias filter.

(ii) To sequentially interleave these N samples inside the sampling interval of $T_s$. In the commutating process, the multiplexed signal is applied to a pulse modulator and the purpose of that is to transform the multiplexed signal into a form suitable for transmission over the common channel. The use of TDM system introduces a bandwidth expansion factor N, because the scheme must squeeze N samples derived from N independent message sources into a time slot equal to one sampling interval.

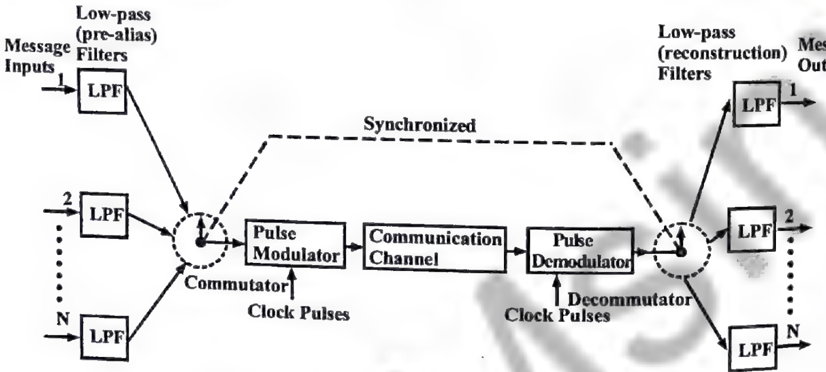


***Fig. 5.11 Block Diagram of TDM System***

At the receiving end of the system, the received signal is applied to a pulse demodulator, that performs the reverse operation of the pulse modulator. The pulse demodulator output is distributed to the appropriate low-pass (reconstruction) filters by means of a demodulator, which operates in synchronism with the commutator in the transmitter.

The TDM system is highly sensitive to dispersion in the common channel, that is to variations of amplitude with frequency. Accurate equalization of both amplitude and phase responses of the channel is necessary to ensure a satisfactory operation of the system.

## NUMERICAL PROBLEMS

***Prob.1. The spectral range of a bandpass signal extends from 10.0 MHz to 10.4 MHz. Determine the minimum sampling rate of signal.***

***Sol.*** In this case the bandwidth is

$$2W = f_H - f_L$$
$$= 10.4 - 10.0 = 0.4 \text{ MHz}$$

The highest frequency is

$$f_c + W = 10.4 \text{ MHz}$$
$$= 26 \times 0.4 \text{ MHz}$$

Because the highest frequency is a integer multiple of the bandwidth 2 W i.e., 0.4 MHz, the lowest sampling rate of the above signal can be obtained as

$$f_s = 4 \text{ W}$$
$$= 2 \times 0.4 \text{ MHz}$$
$$= \mathbf{0.8 \text{ MHz}}$$ **Ans.**

***Prob.2. A signal x(t) = 3 cos(200t) sin(350t) is to be sampled at the smallest allowable rate. What is the sampling rate ?***

***Sol.*** Given,

$$x(t) = 3 \cos(200 t) \sin(350 t)$$
$$= \frac{3}{2}[2\cos(200t)\sin(350t]$$
$$= \frac{3}{2}[\sin(350t+200t)+\sin(350t-200t)]$$
$$= \frac{3}{2}\sin 550t + \frac{3}{2}\sin 150t$$

Hence,

$$\omega_m = 2\pi f_m$$
$$f_m = \frac{\omega_m}{2\pi}$$
$$= \frac{550}{2\pi} = 87.5 \text{ Hz}$$

Sampling rate $= 2 f_m$
$$= 2 \times 87.5 = \mathbf{175 \text{ Hz}}$$ **Ans.**

## PCM, QUANTIZATION ERROR

***Q.10. What is meant by PCM ? State the advantages and disadvantages of PCM when compared with other pulse modulation system.***

***Ans.*** A signal to be quantized before the transmission is sampled also. The main purpose of using quantization is to reduce the effects of noise. A number of messages are multiplexed, by using sampling. The combined operations of sampling and quantizing produce a quantized PAM waveform which is a train of pulses whose amplitudes are restricted to a number of discrete magnitudes.

Generally, the code number is converted, into its representation in binary arithmetic before transmission. The digits of the binary representation of the code number are transmitted as pulses. Thus, the system of transmission is known as ***pulse code modulation (PCM).***

**Advantages –** In pulse amplitude modulation (PAM), pulse-duration modulation (PDM) and pulse-position modulation (PPM), only time is expressed in discrete form, whereas the respective modulation parameters are varied in a continuous manner in accordance with the message. Thus, in these modulation systems, information transmission is accomplished in analog form at discrete times. On the other side, in PCM, the message signal is sampled and the amplitude of each sample is rounded off to the nearest one of the finite set of allowable values, so that both time and amplitude are in discrete form. This permits the message to be transmitted by means of coded electrical signals, thereby distinguishing PCM from all other methods of modulation.

Use of digital representation of analog signals offers the following advantages –

(i) Ruggedness to transmission noise and interference.

(ii) Efficient regeneration of the coded signal along the transmission path.

(iii) The possibility of a uniform format for different kinds of baseband signals.

**Disadvantages –** PCM has following disadvantages as compared to other modulation schemes –

(i) PCM requires very complex encoding and quantizing circuitry.

(ii) PCM requires a large bandwidth compared to analog systems.

***Q.11. Draw the block diagram of PCM system and explain its working.***

***Ans.*** Fig. 5.12 shows the block diagram of a PCM system. The analog signal m(t) is sampled by a sampler and obtained samples are subjected to the operation of quantization. The quantized samples are given to an ***encoder***. The encoder responds to each such sample and generates a unique and identifiable binary pulse or binary level pattern. The pulse pattern happens to have a numerical significance that is same as the order allocated to the quantization levels. But, this feature is not essential. We could have assigned any pulse pattern to any level. At the receiver, we must be able to identify the level from the pulse pattern. Thus, it is obvious that not only does the encoder number the level, it also assigns to it an identification code.

In fig. 5.12, the combination of the quantizer and encoder in the dashed box is called an ***analog-to-digital converter***. The A/D converter accepts an analog signal and replaces it with a succession of code symbols, each symbol having a train of pulses in which each pulse can be interpreted as the representation of a digit in an arithmetic system. Hence, the signal transmitted over the communications channel in a PCM system is referred to as a digitally encoded signal.

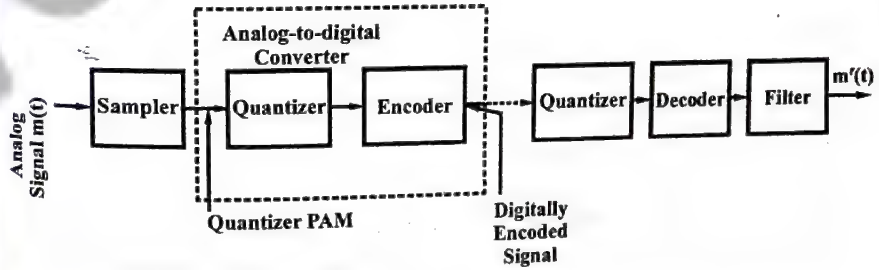


***Fig. 5.12 PCM System***

Then, the digitally encoded signal comes at the receiver or repeater where the first operation to be performed is the separation of the signal from the noise which has been added during the transmission along the channel. The separation of the signal from the noise is possible due to the quantization of the signal. This operation is again an operation of requantization. Thus, the first block in the receiver in fig. 5.12 is termed as quantizer. The feature that eases the burden on this quantizer is that for each pulse interval it has only to make the relatively simple decision of whether a pulse has or has not been received or which of two voltages has occurred.

Then, receiver quantizer in each pulse slot makes an educated and sophisticated estimate and then decides whether a positive pulse or negative pulse was received and transmits its decisions, in the form of a reconstituted or regenerated pulse train, to the decoder. The decoder, also known as a *digital-to-analog (D/A) converter*, does the inverse operation of the encoder. The decoder output is the sequence of quantized multi-level sample pulses. The quantized PAM signal is now reconstituted. Then, it is filtered to reject any frequency components lying outside of the baseband. The final output signal m'(t) is similar to the input m(t) except for quantization noise, and the occasional error in yes-no decision making at the receiver owing to the presence of channel noise.

***Q.12. What are the processes involved in PCM ? State its advantages and disadvantages when compared with other pulse modulation system.***
***(R.G.P.V., Dec. 2002)***

*Ans.* **PCM** – Refer the ans. of Q.10.

The important aspects of PCM are illustrated in fig. 5.13 and table 5.1. It is supposed that the analog message signal x(t) is extended to the range from – 4 to + 4 volts. The step size between quantization levels is fixed at 1 volt. Eight levels of quantization are used. These are situated at – 3.5, – 2.5, –1.5, ......, + 3.5 volts. At – 3.5 volts, allocate the code number 0, similarly level at –2.5 volts is allocated code number 1, proceeds in this way upto the level at + 3.5 volts, which is assigned the code number 7.

The sample value, the nearest quantization level, and the code number and its binary representation corresponding to each sample is shown in table 5.1. We would transmit the sample values 1.3, 3.6, 2.3 etc., when analog signal is being transmitted. We would transmit the quantized sample values 1.5, 3.5, 2.5 etc., when quantized signal is being transmitted. The binary representations 101, 111, 110 etc. are transmitted in binary pulse code modulation.

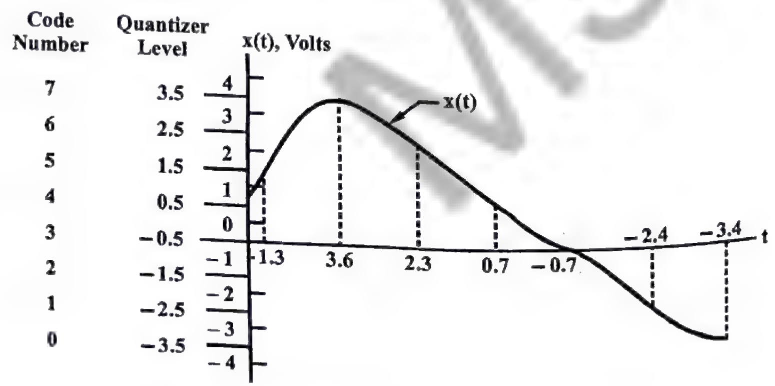


*Fig. 5.13 Process Involved in Binary PCM*

**Table 5.1**

| Sample Value of an Analog Signal | 1.3 | 3.6 | 2.3 | 0.7 | – 0.7 | – 2.4 | – 3.4 |
|---|---|---|---|---|---|---|---|
| Nearest Quantizer Level | 1.5 | 3.5 | 2.5 | 0.5 | – 0.5 | – 2.5 | – 3.5 |
| Code Number | 5 | 7 | 6 | 4 | 3 | 1 | 0 |
| Binary Code | 101 | 111 | 110 | 100 | 011 | 001 | 000 |

**Advantages and Disadvantages** – Refer the ans. of Q.10.

***Q.13. What do you understand by quantization ?***

***Ans. Quantization*** is the process of converting a discrete-time continuous amplitude signal such as the sampled version of an analog signal into a discrete-amplitude discrete-time signal. This is accomplished by approximating the amplitude of each sample value to the nearest value from a set of predetermined discrete amplitude levels known as ***quantization levels***.

***Q.14. Explain the need of quantization. How is it done ? What should be the limitation for selecting the step size ?***

***Ans.*** The aim of quantization in a practical system can be appreciated from the following discussion. A continuous signal, like voice in the case of audio transmission or picture in the case of video transmission, has a continuous range of amplitudes. If these signals are sampled, these have a continuous amplitude range which results in an infinite number of amplitude levels in a given finite amplitude range over which the signals vary. Since

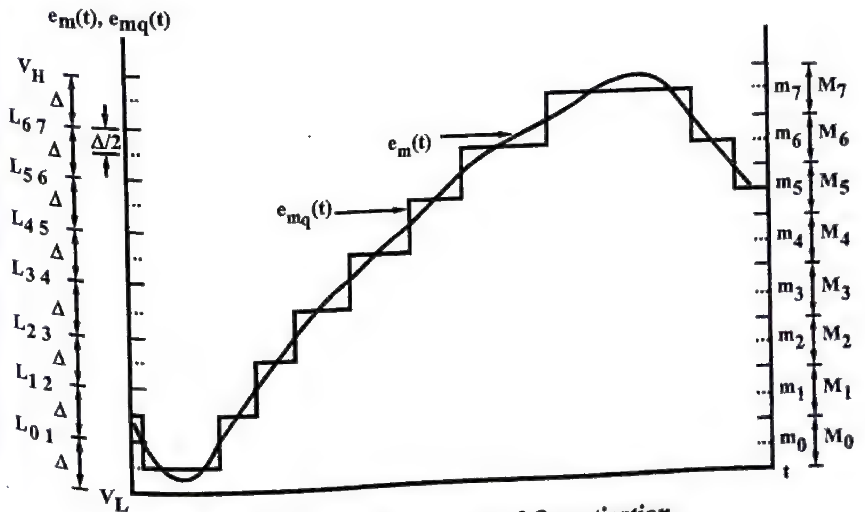


*Fig. 5.14 The Operation of Quantization*

human ear (in case of audio) and eye (in case of video) cannot find small finite differences in amplitude (intensity) of the signals, it is logical to consider that the original continuous amplitude signal can be approximated by a discrete amplitude signal constructed using discrete amplitude chosen on a minimum error basis from an available set. The existance of a finite number of discrete amplitude levels is a basic condition for PCM. Hence, it is obvious that by selecting the discrete-amplitude levels sufficiently close, we can make the approximated signal indistinguishable from the original signal for all practical purposes. We can take the example of commercial colour television for understanding. While 64 levels provide only fairly good colour TV performance, use of 256 levels provides high quality commercial colour TV performance. These results are also valid for quantizing voice signals.

Fig. 5.14 illustrates the operation of quantization. Now assume a signal $e_m(t)$ whose excursion is confined in the range from $V_L$ to $V_H$. This total range $V_H - V_L$ is divided into M equal intervals each of size Δ. Accordingly, the Δ is called the step size. It is given by the expression –

$$\Delta = \frac{V_H - V_L}{M} \qquad \text{...(i)}$$

In fig. 5.14, 8 equal intervals are taken, that is, M = 8 levels. In the centre of every steps, we locate quantization levels denoted by $m_0$, $m_1$, $m_2$, $m_3$,...., $m_7$. Quantized signal $e_{mq}$ (t) is generated by adopting the following scheme. Whenever $e_m(t)$ is in the range $M_0$, $e_{mq}(t)$ maintains the constant level $m_0$. Whenever it is in the range $M_1$, $e_{mq}(t)$ maintains constant level $m_1$ and so on. Therefore, quantized signal $e_{mq}(t)$ will at all times be found at one of the levels $m_0$, $m_1$, $m_2$, $m_3$,...., $m_7$.

The transition in $e_{mq}(t)$ from $m_0$ to $m_1$, $m_1$ to $m_2$, $m_2$ to $m_3$ etc. is made abruptly when $e_m(t)$ passes the corresponding transition level ($L_{01}$, $L_{12}$) that is midway between the states $m_0$ and $m_1$, $m_1$ and $m_2$, and so on. Therefore, at every instant of time $e_{mq}(t)$ does not change at all with time or it makes discrete quantized jumps of step size Δ. It should be noted that quantization levels are separated by an amount Δ. On the other side, the separation of extreme levels $V_L$ and $V_H$ each from their nearest quantization level is only Δ/2. It must also be noted that at any instant, the quantization error $e_m(t) - e_{mq}(t)$ has a magnitude that is equal to or less than Δ/2. Therefore, it can be concluded that the quantized signal is an approximation to the original signal and this approximation can be improved by decreasing the size of the steps, thereby increasing the number of permitted levels.

By using the process of signal quantization, the effect of noise cannot be stopped completely but can be reduced significantly. Further by reducing the spacing between the repeaters, we can reduce the attenuation suffered by the quantized signal $e_{mq}(t)$, due to which the relative noise power is effectively decreased and thus the probability $P_q$ of an error in the level. There is also a way to reduce the $P_q$ significantly, by increasing the step size. But, increasing step size results an increased discrepancy between the actual signal $e_m(t)$ and the quantized signal $e_{mq}(t)$. The difference $e_m(t) - e_{mq}(t)$ is known to as ***quantization noise***. Hence, the recovered signal is not a perfect replica of the transmitted signal $e_m(t)$. The difference between them is because of error caused by additive noise and quantization noise.

***Q.15. What is quantization error ? How does it depend upon the step size and how it can be reduced ?***

***Or***

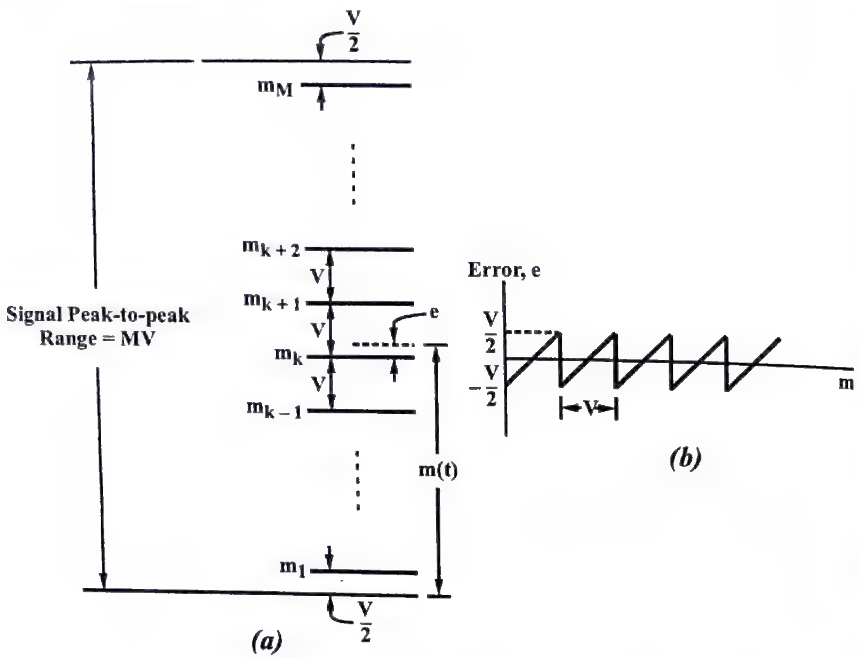
***Write short note on quantization error.*** ***(R.G.P.V., Nov. 2018)***

***Ans.*** **Quantization Error –** The quantized signal and the original signal from which it was derived differ from one another in a random manner. This difference or error may be viewed as a noise due to the quantization process and is known as ***quantization error*** or ***quantization noise***.

**Relation between Quantization Error and Step Size –** The original and quantized signals are different from one another in a random fashion, from which it was derived. The quantization error is known as the difference or error which can be viewed as a noise because of quantization process. Now, we compute the mean-square quantization error $(\overline{e^2})$. where e denotes the difference between the quantized and original signal voltages.

The message signal m(t) total peak-to-peak range is divided into M equal voltage intervals, every signal has V volts magnitude. As shown in fig. 5.15 (a), at the center of every voltage interval we situated a quantization level $m_1$, $m_2$, $m_3$,....., $m_M$.

Fig. 5.15 (a) shows a voltage range over which a signal m(t) makes excursions is divided into M quantization ranges each of size V. The quantization levels are situated at the center of the range and (b) represents the error voltage e(t) as a function of the instantaneous value of the signal m(t).



*(a)* *(b)*

*Fig. 5.15*

Consider f(m) dm be the probability that m(t) exists in the voltage range m – dm/2 to m + dm/2. So, the mean-square quantization error is

$$\overline{e^2} = \int_{m_1-V/2}^{m_1+V/2} f(m)\,(m-m_1)^2\,dm + \int_{m_2-V/2}^{m_2+V/2} f(m)\,(m-m_2)^2\,dm + \ldots \quad \ldots(i)$$

Normally the probability density function f(m) of the message signal m(t) will definitely not be constant. However assume that the number M of quantization is large, so that the step size V is small in comparison with the peak-to-peak range of the message signal. In this case, it is definitely reasonable to make the approximation that f(m) is constant within every quantization range. Then, we set $f(m) = f^{(1)}$ a constant in the first term of equation (i). In the second term $f(m) = f^{(2)}$, etc. Now we can remove $f^{(1)}$, $f^{(2)}$ etc., from innerside of the integral sign. When we make the substitution $x \equiv m - m_k$ equation (i) becomes

$$\overline{e^2} = (f^{(1)} + f^{(2)} + \ldots)\int_{-V/2}^{V/2} x^2 dx = (f^{(1)} + f^{(2)} + \ldots)\frac{V^3}{12}$$

$$= (f^{(1)}V + f^{(2)}V + \ldots)\frac{V^2}{12} \quad \ldots(ii)$$

Now $f^{(1)}V$ is the probability that the signal voltage m(t) will be in the first quantization range, $f^{(2)}V$ is the probability that m is in the second quantization range etc. Thus the sum of terms in the parentheses in equation (ii) has a total value of unity. Therefore, the mean-square quantization error is

$$\overline{e^2} = \frac{V^2}{12} \quad \ldots(iii)$$

***Q.16. Derive the expression for quantization noise (or error).***

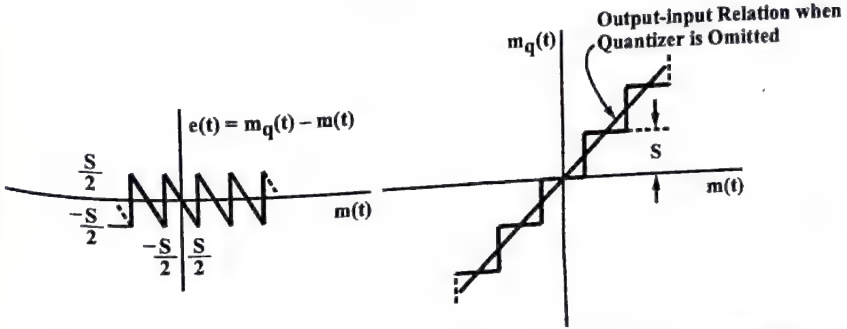
***Ans.*** The equation of sampled quantization error waveform is given below –

$$e_s(t) = e(t) I \sum_{k=-\infty}^{\infty} \delta(t - kT_s) \quad \ldots(i)$$

It is noted that when the sampling rate is chosen to be Nyquist rate for the baseband signal m(t), the sampling rate will be inadequate to permit reconstruction of the error signal e(t) from its samples $e_s(t)$. That such is the case is readily apparent from fig. 5.16. Fig 5.16 (b) shows the error waveform e(t) as a function of m(t) while fig. 5.16 (a) shows the relationship between $m_q(t)$ and m(t). Amount S is used to separates the quantization levels. We found in fig. 5.16 (b) that e(t) executes a complete cycle and exhibits an abrupt discontinuity every time m(t) makes an excursion of amount S. Thus the spectral range of e(t) extends far beyond the bandlimit $f_M$ of m(t).

To determine the quantization noise output power $N_q$, we need the power spectral density of the sampled quantization error $e_s(t)$, given in equation (i) Since $\delta(t - kT_s) = 0$ except when $t = kT_s$, $e_s(t)$ may be written

$$e_s(t) = I \sum_{k=-\infty}^{\infty} e(kT_s)\,\delta(t - kT_s) \quad \ldots(ii)$$



***(a) Plot of e(t) as a Function of m(t)*** ***(b) Plot of $m_q(t)$ as a Function of m(t)***

***Fig. 5.16***

The waveform of equation (ii) have a sequence of impulses of area (strength) A = e(kT$_s$)I occurring at intervals T$_s$. The quantization error is the quantity e(kT$_s$) at the sampling time and is a random variable.

We find that the power spectral density $G_{e_s}(f)$ of the sampled quantization error is

$$G_{e_s}(f) = \frac{I^2}{T_s}\overline{e^2(kT_s)} \qquad \text{...(iii)}$$

We found that if the quantization levels are separated by amount S, then the quantization error is given by

$$\overline{e^2(t)} = \frac{S^2}{12} \qquad \text{...(iv)}$$

Equation (iii) involves $\overline{e^2(kT_s)}$ rather than $\overline{e^2(t)}$. However, since the probability density of e(t) does not depend on time, the variance of e(t) is equal to the variance of e(t = kT$_s$). Hence,

$$\overline{e^2(t)} = \overline{e^2(kT_s)}$$

$$= \frac{S^2}{12} \qquad \text{...(v)}$$

From equations (iii), (iv) and (v), we get

$$G_{e_s}(f) = \frac{I^2S^2}{T_s12} \qquad \text{...(vi)}$$

Finally, the quantization noise N$_q$ is, from equation (vi)

$$N_q = \int_{-f_M}^{f_M} G_{e_s}(f)\,df$$

$$= \frac{I^2S^2}{T_s12}2f_M$$

$$= \frac{I^2S^2}{T_s^2 12} \qquad \left(\because 2f_M = \frac{1}{T_s}\right) \qquad \text{...(vii)}$$

Of more interest than the quantization noise given in equation (vii) is the signal-to-quantization noise ratio.

***Q.17. Derive the expression of signal-to-quantization noise ratio.***

***Ans.*** Fig. 5.17 shows the quantization error as a function of input voltage. This can be seen from the figure that the quantization error can lie between ± ΔV/2, and assuming it has a uniform probability density distribution, it can be shown that the mean-square quantization error is given by the expression –

$$E_{nq}^2 = \frac{(\Delta V)^2}{12} \qquad \text{...(i)}$$

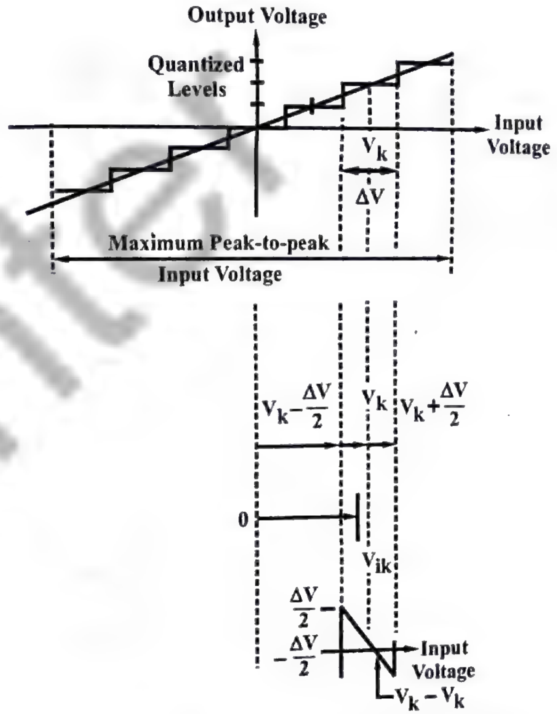


*Fig. 5.17 Quantization Error*

For a total number of L levels, the peak-to-peak signal range is $\pm\frac{L\Delta V}{2}$, and for a signal which has a uniform probability density distribution within this range, the mean-square, signal voltage is given by the expression

$$E_s^2 = \frac{(L\Delta V)^2}{12} \qquad \text{...(ii)}$$

Therefore, the signal-to-quantization noise ratio is obtained as

$$\left(\frac{S}{N}\right)_q = \frac{E_s^2}{E_{nq}^2} = L^2 \qquad \text{...(iii)}$$

Equation (iii) shows that to maintain a high (S/N)$_q$ ratio, the number of steps should be high. In terms of the number of bits per code word n, L = 2$^n$, and hence

$$\left(\frac{S}{N}\right)_q = 2^{2n} \qquad \text{...(iv)}$$

In decibels, we have

$$\left[\left(\frac{S}{N}\right)_q\right]_{dB} = 10\log_{10}\left(\frac{S}{N}\right)_q = 10\log_{10}2^{2n} \simeq 6n$$

## NUMERICAL PROBLEMS

***Prob.3. 48 telephone channels each limited to 3.4 kHz are to be time division multiplexed by using PCM. Calculate the bandwidth of the PCM system for 128 quantization levels and 8 kHz sampling frequency.***

***(R.G.P.V., June 2003, Nov./Dec. 2007)***

***Sol.*** Given,

Number of channels (n) = 48

Quantization levels (M) = 128

Since, $2^N = 128$

$2^N = 2^7$

$\therefore$ $N = 7$

Sampling frequency $(2f_m) = 8$ kHz $= 8 \times 10^3$ Hz

Now, the bandwidth of the PCM system is calculated as

$$BW = [n(N+1)+1]\,2f_m \text{ Hz}$$
$$= [48(7+1)+1] \times 8 \times 10^3 \text{ Hz}$$
$$= \mathbf{3.08\ MHz}$$ **Ans.**

## INTRODUCTION TO BPSK & BFSK MODULATION SCHEMES, SHANNON'S THEOREM FOR CHANNEL CAPACITY

***Q.18. What is meant by phase shift keying (PSK) ?***

***Ans.*** PSK (phase-shift keying) is an M-ary digital modulation scheme similar to conventional phase modulation except with PSK the input is a binary digital signal and there are a limited number of output phases possible. Before modulating the carrier, the input binary information is encoded into groups of bits. In a group, the number of bits ranges from 1 to 12 or more. The number of output phases is defined by M and determined by the number of bits in the group(n). Here, M is defined as –

$$M = 2^N$$

where N = Number of bits necessary

M = Number of conditions, levels,or combinations possible with N bits.

Binary phase-shift keying (BPSK) is the simplest form of PSK, where N = 1 and M = 2. With BPSK, two phases ($2^1 = 2$) are possible for the carrier. One phase represents a logic 1, and the other phase represents a logic 0. When the input digital signal changes state (i.e., from 1 to 0 or from 0 to 1), the phase of the output carrier shifts between two angles that are separated by 80°.

***Q.19. Draw and explain the block diagram of BPSK transmitter and receiver.***

***Ans.*** The block diagram of BPSK transmitter and receiver are as follows –

**BPSK Transmitter –** A simplified block diagram of a BPSK transmitter is shown in fig. 5.18. The balanced modulator acts as a phase reversing switch. The carrier is transferred to the output either in phase or 180° out of phase with the reference carrier oscillator, depending on the logic condition of the digital output.

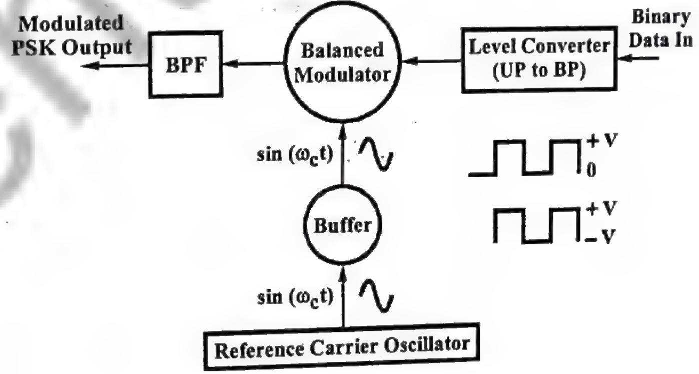
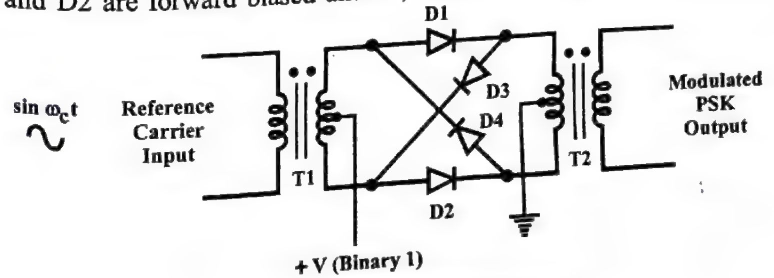
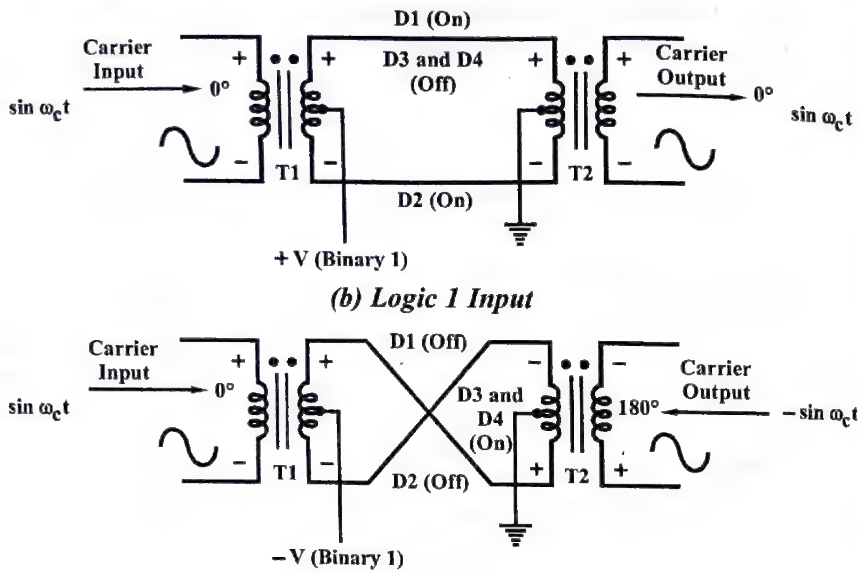


***Fig. 5.18 BPSK Transmitter***

The schematic diagram of a balanced ring modulator is shown in fig. 5.19. The balanced modulator consists of two inputs – a carrier that is in phase with the reference oscillator and the binary digital data. The digital input voltage must be much greater than the peak carrier voltage to operate the balanced modulator properly. This ensures that the digital input controls the on/off state of diodes D1 to D4. When the binary input is a logic 1 (positive voltage), diodes D1 and D2 are forward biased and on, while diodes D3 and D4 are reverse



***(a) Balanced Ring Modulator***



*(b) Logic 1 Input*

*(c) Logic 0 Input*

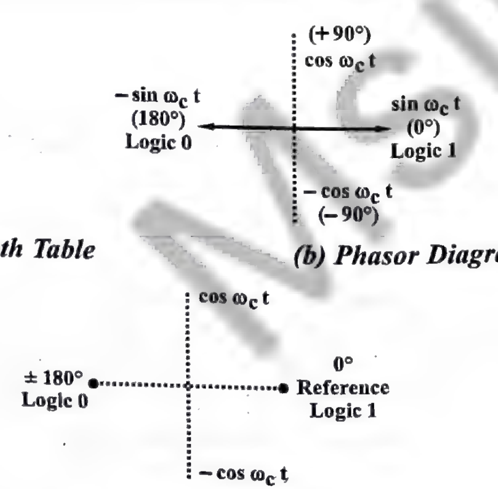
*Fig. 5.19*

biased and off [refer fig. 5.19 (b)]. With the polarities shown, the carrier voltage is developed across transformer T2 in phase with the carrier voltage across T1. As a result, the output signal is in phase with the reference oscillator.

When the binary input is a logic 0 (negative voltage), diodes D1 and D2 are reverse biased and off, while diodes D3 and D4 are forward biased and on [refer fig. 5.19 (c)]. Consequently, the carrier voltage is developed across transformer T2 180° out of phase with the carrier voltage across T1. As a result, the output signal is 180° out of phase with the reference oscillator.

| Binary Input | Output Phase |
|---|---|
| Logic 0 | 180° |
| Logic 1 | 0° |

*(a) Truth Table*



*(b) Phasor Diagram*

*(c) Constellation Diagram*

*Fig. 5.20 BPSK Modulator*

Fig. 5.20 depicts the truth table, phasor diagram and constellation diagram for a BPSK modulator. A constellation diagram, sometimes also known as a signal state-space diagram, is similar to a phasor diagram except that the entire phasor is not drawn. In a constellation diagram, only the relative positions of the peaks of the phasors are shown.

**BPSK Receiver** – The block diagram of a BPSK receiver is shown in fig. 5.21. The input signal may be + sin $\omega_c$t or – sin $\omega_c$t. The coherent carrier recovery circuit detects and reproduces a carrier signal which is both frequency and phase coherent with the original transmit carrier. The balanced modulator is a product detector; the output is the product of the two inputs (the BPSK signal and the recovered carrier). The low-pass filter (LPF) separates the recovered binary data from the complex demodulated signal.

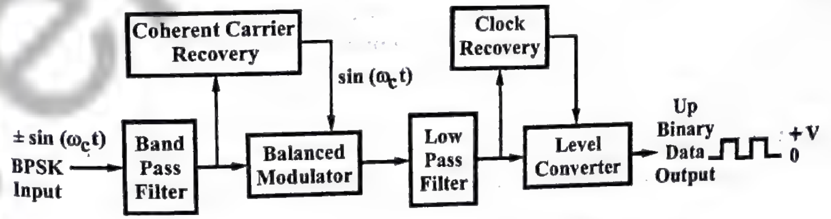


*Fig. 5.21 Block Diagram of a BPSK Receiver*

The mathematical description of the demodulation process is as follows – For a BPSK input signal of + sin $\omega_c$t (logic 1), the output of the balanced modulator is given as –

$$\text{Output} = (\sin \omega_c t)(\sin \omega_c t) = \sin^2\omega_c t \qquad \text{...(i)}$$

or

$$\sin^2\omega_c t = \frac{1}{2}(1-\cos 2\omega_c t) = \frac{1}{2} - \underbrace{\frac{1}{2}\cos 2\omega_c t}_{\text{Filtered out}}$$

$$\text{Leaving output} = +\frac{1}{2}V = \text{Logic 1}$$

It can be noted that the output of the balanced modulator contains a positive voltage [+ (1/2)V] and a cosine wave at twice the carrier frequency ($2\omega_c$). The LPF has a cut-off frequency much lower than $2\omega_c$ and hence blocks the second harmonic of the carrier and passes only the positive constant component. A positive voltage represents a demodulated logic 1.

With a BPSK input signal of – sin $\omega_c$t (logic 0), the output of the balanced modulator is given as –

$$\text{Output} = (-\sin\omega_c t)(\sin\omega_c t) = -\sin^2\omega_c t$$

or

$$-\sin^2\omega_c t = -\frac{1}{2}(1-\cos 2\omega_c t) = -\frac{1}{2} + \underbrace{\frac{1}{2}\cos 2\omega_c t}_{\text{Filtered out}}$$

$$\text{Leaving output} = -\frac{1}{2}\text{V} = \text{Logic } 0$$

The balanced modulator's output consists of a negative voltage [– (1/2)V] and a cosine wave at twice the carrier frequency ($2\omega_c$). Again, the LPF blocks the second harmonic of the carrier and passes only the negative constant component. A negative voltage indicates a demodulated logic 0.

***Q.20. Explain in detail differential PSK.***

***Ans.*** To eliminate the need for phase synchronization of coherent receiver with PSK, a differential encoding system is used in this modified scheme. The digital information content of the binary data is encoded in terms of signal transitions. The symbol 0 may be used to represent transition in a given binary sequence and symbol 1 to indicate no transition. This new signaling technique that combines differential encoding with phase shift keying is called differential phase shift keying (DPSK).

A schematic arrangement for generating DPSK signal is shown in fig. 5.22.

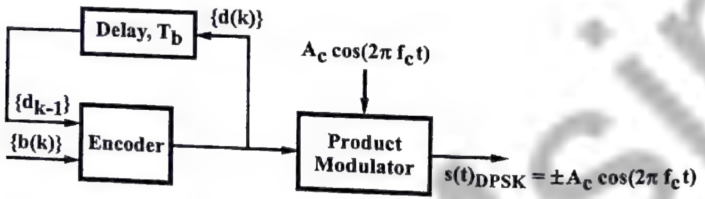


*Fig. 5.22 Scheme for Generating DPSK Signals*

The data stream b(t) is applied to the input of the encoder. The output of the encoder is applied to one input of the product modulator. To the other input of this product modulator a sinusoidal carrier of fixed amplitude and frequency is applied. The relationship between the binary sequence and its differentially encoded version is illustrated in table 5.2 for a assumed data sequence 00100100111.

From the table it has been assumed that the encoding has been done in such a way that transition in the given binary sequence with respect to the previous encoded bit is represented by a symbol 0 and no transition by symbol 1. An extra bit (symbol 1) has been arbitrarily added as an initial bit.

**Table 5.2 Differentially Encoded Sequences with Phase**

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Binary data {b(k)} | | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| Differentially encoded data {d(k)} | 1* | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| Phase of DPSK | 0 | π | 0 | 0 | π | 0 | 0 | π | 0 | 0 | 0 |
| Shifted differentially encoded data $\{d_{k-1}\}$ | | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| Phase of shifted DPSK | | 0 | π | 0 | 0 | π | 0 | 0 | π | 0 | 0 |
| Phase comparison output | | - | - | + | - | - | + | - | - | + | + |
| Detected binary sequence | | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| *arbitrary starting reference bit | | | | | | | | | | | |

This is necessary to determine the encoded sequence.

For detection of the differentially encoded PSK (DPSK), we may use the receiver arrangement shown in fig. 5.23.

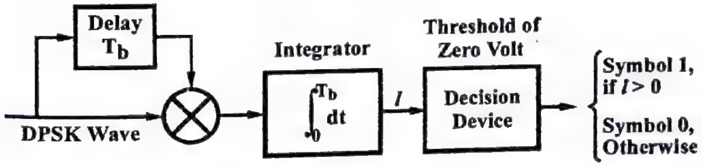


*Fig. 5.23 Receiver for the Detection of DPSK Signal*

The received DPSK signal is applied to one input of the multiplier. To the other input of the multiplier a delayed version of the received DPSK signal by the time interval $T_b$ is applied. The delayed version of the received DPSK signal is shown in the 4th row of the table. The output of the difference is proportional to cos ϕ, where ϕ is the difference between the carrier phase angle of the received DPSK signal and its delayed version measured in the same bit interval. The phase angle of the DPSK signal and its delayed version are in 3rd and 5th rows respectively. The phase difference between the two sequences for each bit interval is used to determine the sign of the phase comparator output. When $\phi = 0$, the integrator output is positive whereas when $\phi = \pi$, the integrator output is negative. By comparing the integrator output with a decision level of zero volt, the decision device can reconstruct the binary sequence by assigning a symbol 0 for negative output and a symbol 1 for positive output. The reconstructed binary data is shown in the last row of the table. It is thus seen that in the absence of noise the receiver can reconstruct the transmitted binary data exactly. DPSK may be viewed as a non-coherent version of PSK.

The main merit of DPSK is that no synchronous carrier is needed at the receiver. Its demerit is that error generates in pairs, since single errorneous bit affects the two successive bit intervals. Thus the error rate of DPSK is higher than PSK.

***Q.21. What is meant by QPSK ?***

***Ans.*** QPSK (quadrature PSK) is an M-ary encoding scheme where N = 2 and M = 4. In QPSK, four output phases are possible for a single carrier frequency. Since there are four output phases, there must be four different input conditions. Since the digital input to a QPSK modulator is a binary (base 2) signal, to generate four different input combinations, the modulator needs more than a single input bit to determine the output condition. With two bits, there are four possible conditions – 00, 01, 10 and 11. Thus, with QPSK, the binary input data are combined into groups of two bits, known as dibits. In the modulator, each dibit code produces one of the four possible output phases (+ 45°, + 135°, – 45°, and – 135°). So, a single output change takes place for each two-bit dibit clocked into the modulator, and the rate of change at the output (baud) is equal to one-half the input bit rate, i.e., two input bits generate one output phase change.

***Q.22. Draw the block diagram of QPSK transmitter and receiver and explain their operation.***

***Ans.*** **QPSK Transmitter** – Fig. 5.24 shows a block diagram of a QPSK modulator. Two bits are clocked into the bit splitter. After both bits have been serially inputted, they are simultaneously parallel outputted. One bit is directed to the I channel and the other to the Q channel. The I bit modulates a carrier that is in phase with the reference oscillator, and the Q-bit modulates a carrier which is 90° out of phase or in quadrature with the reference carrier.

It is worthnoticed that once a dibit has been split into the I and Q channels, the operation is the same as in a BPSK modulator. A QPSK modulator is essentially two BPSK modulators combined in parallel. Again, for a logic 1 = + 1 V and a logic 0 = – 1 V, two phases are possible at the output of the I balanced modulator (+ sin $\omega_c t$ and – sin $\omega_c t$), and two phases are possible at the output of the Q balanced modulator (+ cos $\omega_c t$ and – cos $\omega_c t$). As the linear summer combines the two quadrature (90° out of phase) signals, there are four possible resultant phasors given by these expressions : + sin $\omega_c t$ + cos $\omega_c t$, + sin $\omega_c t$ – cos $\omega_c t$, – sin $\omega_c t$ + cos $\omega_c t$ and – sin $\omega_c t$ – cos $\omega_c t$.

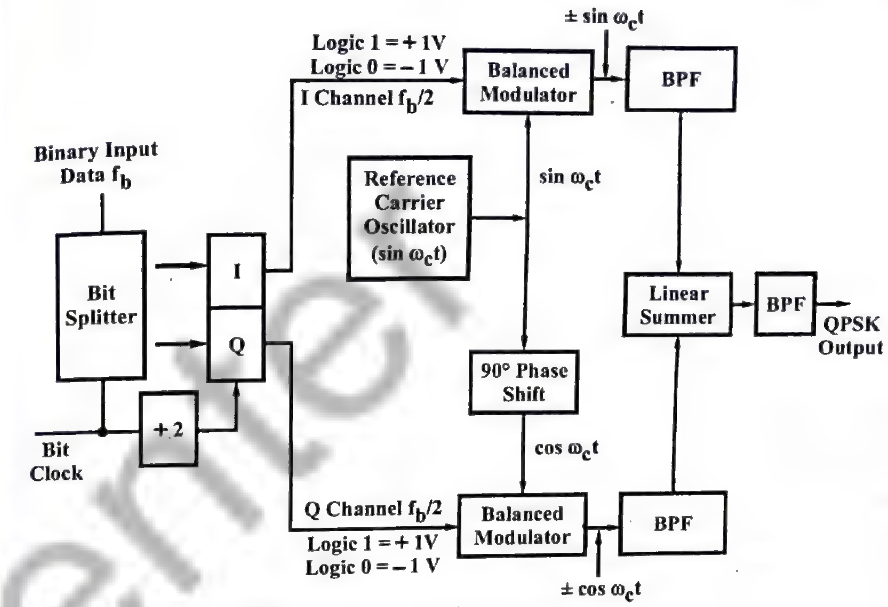
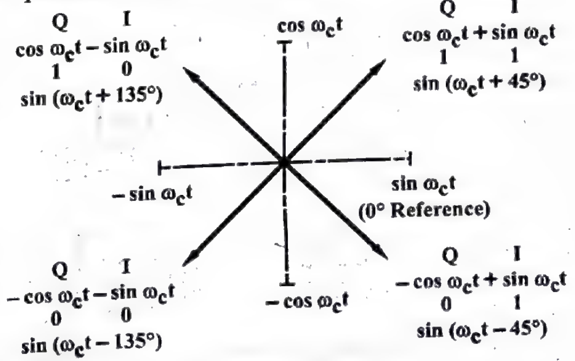


*Fig. 5.24 QPSK Modulator*

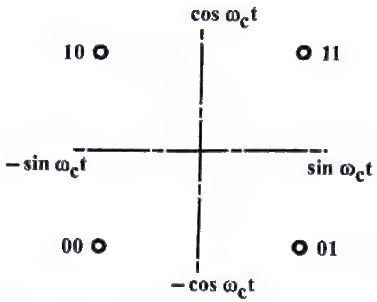
It is noted from fig. 5.25 (b) and (c) that with QPSK, each of the four possible output phasors has exactly the same amplitude. So, the binary information must be encoded entirely in the phase of the output signal. This constant amplitude characteristic is the most important characteristic of PSK which distinguishes it from QAM. It is also observed from fig. 5.25 (b) that the angular separation between any two adjacent phasors in QPSK is 90°. So, a QPSK signal can

| Binary Input | | QPSK Output Phase |
|---|---|---|
| Q | I | |
| 0 | 0 | – 135° |
| 0 | 1 | – 45° |
| 1 | 0 | + 135° |
| 1 | 1 | + 45° |

*(a) Truth Table*



*(b) Phasor Diagram*



*(c) Constellation Diagram*

***Fig. 5.25 QPSK Modulator***

undergo almost a + 45° or – 45° shift in phase during transmission and still retain the correct encoded information when demodulated at the receiver. The output phase-versus-time relationship for a QPSK modulator is shown in fig. 5.26.

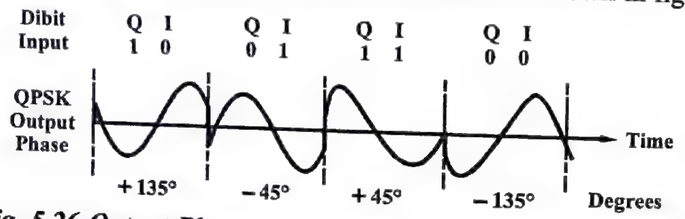


***Fig. 5.26 Output Phase-versus-time Relationship for a QPSK***

**QPSK Receiver** – Fig. 5.27 shows the block diagram of a QPSK receiver. The power splitter directs the input QPSK signal to the I and Q product detectors and the carrier recovery circuit. The carrier recovery circuit regenerates the original transmit carrier oscillator signal. The recovered carrier must be frequency and phase coherent with the transmit reference carrier. In the I and Q product detectors, the QPSK signal is demodulated and the original I and Q data bits are produced. The outputs of the product detectors are applied to the bit combining circuit, where they are converted from parallel I and Q data channels to a single binary output data stream.

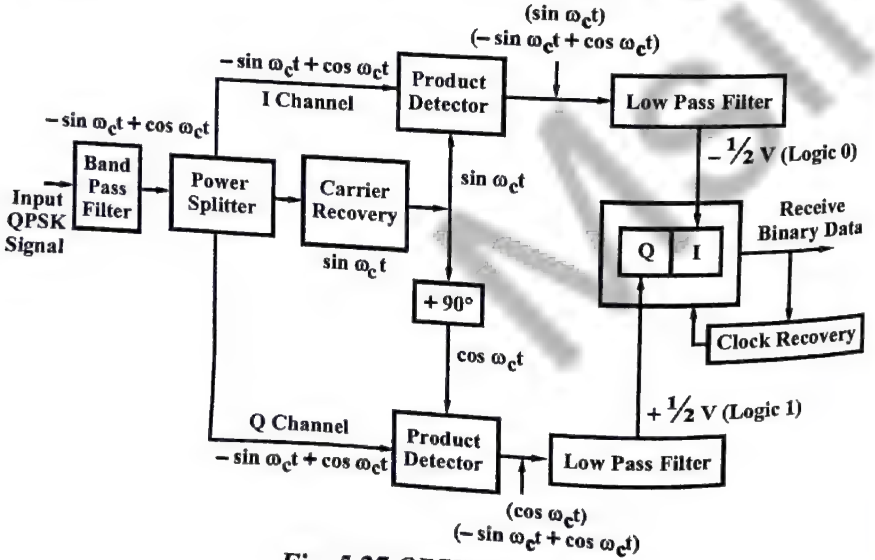


***Fig. 5.27 QPSK Receiver***

The incoming QPSK signal may be any one of the four possible output phases. To explain the demodulation process, let us consider the incoming QPSK signal be – sin $\omega_c$t + cos $\omega_c$t. In mathematical terms, the demodulation process is as follows –

The one input to the I product detector is the receive QPSK signal (– sin $\omega_c$t + cos $\omega_c$t), and the other input is the recovered carrier (sin $\omega_c$t). Then, the output of the I product detector is obtained as

$$I = \underbrace{(-\sin\omega_c t + \cos\omega_c t)}_{\text{QPSK input signal}}\,\underbrace{(\sin\omega_c t)}_{\text{Carrier}}$$

$$= (-\sin\omega_c t)(\sin\omega_c t) + (\cos\omega_c t)(\sin\omega_c t)$$

$$= -\sin^2\omega_c t + (\cos\omega_c t)(\sin\omega_c t)$$

$$= -\frac{1}{2}(1-\cos 2\omega_c t) + \frac{1}{2}\sin 2\omega_c t$$

$$I = -\frac{1}{2} + \underbrace{\frac{1}{2}\cos 2\omega_c t}_{\text{Filtered out}} + \underbrace{\frac{1}{2}\sin 2\omega_c t}_{\text{Equals 0}} = -\frac{1}{2}\text{V(logic 0)}$$

Now, again, one of the inputs to the Q product detector is the receive QPSK signal (– sin $\omega_c$t + cos $\omega_c$t), and the other input is the recovered carrier shifted 90° in phase (cos $\omega_c$t). Then, the output of the Q product modulator is obtained as

$$Q = \underbrace{(-\sin\omega_c t + \cos\omega_c t)}_{\text{QPSK input signal}}\,\underbrace{(\cos\omega_c t)}_{\text{Carrier}}$$

$$= \cos^2\omega_c t - (\sin\omega_c t)(\cos\omega_c t)$$

$$= \frac{1}{2}(1+\cos 2\omega_c t) - \frac{1}{2}\sin 2\omega_c t$$

$$Q = \frac{1}{2} + \underbrace{\frac{1}{2}\cos 2\omega_c t}_{\text{Filtered out}} - \underbrace{\frac{1}{2}\sin 2\omega_c t}_{\text{Equals 0}} = \frac{1}{2}\text{V(logic 1)}$$

***Q.23. Discuss the frequency shift keying (FSK) in breif.***

***Ans.*** Frequency shift keying (FSK) is a form of constant-amplitude angle modulation similar to standard frequency modulation (FM) except the modulating signal is a binary signal that varies between two discrete voltage levels rather than a continuously changing analog waveform. Consequently, FSK is sometimes known as binary FSK (BFSK).

The general expression for FSK is given as

$$v_{fsk}(t) = V_c \cos [2\pi \{f_c + v_m(t)\, \Delta f\ \}\ t] \qquad ...(i)$$

where, $v_{fsk}(t)$ = Binary FSK waveform

$v_m(t)$ = Binary input (modulating) signal

$V_c$ = Peak analog carrier amplitude

$f_c$ = Analog carrier center frequency

$\Delta f$ = Peak change (shift) in the analog carrier frequency.

It is observed from equation (i) that the peak shift in the carrier frequency ($\Delta f$) is proportional to the amplitude of the binary input signal [$v_m(t)$], and the direction of the shift is determined by the polarity. Modulating signal is a normalized binary waveform where a logic 1 = + 1 V and a logic 0 = – 1 V. Hence, for a logic 1 input, $v_m(t) = +1$, equation (i) becomes in the form as

$$v_{fsk}(t) = V_c \cos [2\pi (f_c + \Delta f)\ t]$$

For a logic 0 input, $v_m(t) = -1$, equation (i) can be rewritten in the form as

$$v_{fsk}(t) = V_c \cos [2\pi (f_c - \Delta f)\ t]$$

With binary FSK, carrier center frequency ($f_c$) is shifted (deviated) up and down in the frequency domain by the binary input signal as depicted in fig. 5.28.

When the binary input signal changes from a logic 0 or to a logic 1 and vice-versa, the output frequency shifts between two frequencies – a mark or logic 1 frequency ($f_m$), and a space or logic 0 frequency ($f_s$). The mark and space frequencies are separated from the carrier frequency by the peak frequency deviation ($\Delta f$) and from each other by $2\Delta f$.

Frequency deviation in FSK is defined as the difference between either the mark or space frequency and the center frequency or half the difference between the mark and space frequencies. Frequency deviation is shown in fig. 5.28 and its mathematical expression is given as

$$\Delta f = \frac{|f_m - f_s|}{2}$$

where $\Delta f$ = Frequency deviation (hertz)

$|f_m - f_s|$ = Absolute difference between the mark and space frequencies (hertz).

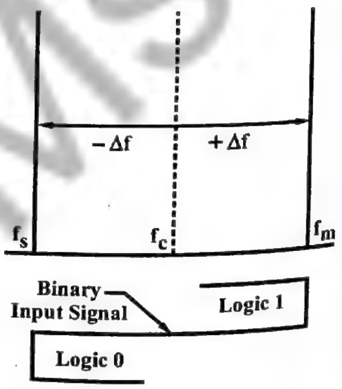


*Fig. 5.28 FSK in the Frequency Domain*

***Q.24. Explain BFSK with block diagram of transmitter and receiver.***

***Ans.*** **BFSK Transmitter** – A simplified binary FSK modulator is shown in fig. 5.29, which is very similar to a conventional FM modulator and is very often a voltage-controlled oscillator (VCO). The center frequency is selected such that it falls halfway between the mark and space frequencies. A logic 1 input shifts the VCO output to the mark frequency l and a logic 0 input shifts the VCO output to the space frequency. Consequently, as the binary input signal changes back and forth between logic 1 and logic 0 conditions, the VCO output shifts or deviates back and forth between the mark and space frequencies.

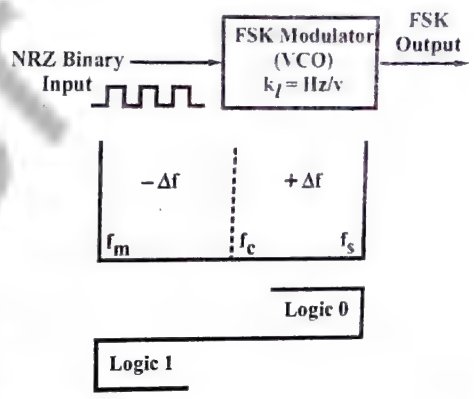


*Fig. 5.29 BFSK Modulator*

With a binary FSK modulator, $\Delta f$ is the peak frequency deviation of the carrier and is equal to the difference between the carrier rest frequency and either the mark or the space frequency (or half the difference between the mark and space frequencies). A VCO-FSK modulator can be operated in the sweep mode where the peak frequency deviation is simply the product of the binary input voltage and the deviation sensitivity of the VCO. The mathematical expression of frequency deviation for the sweep mode of modulation is given as

$$\Delta f = v_m(t)\ k_l$$

where $\Delta f$ = Peak frequency deviation (hertz)

$v_m(t)$ = Peak binary modulating-signal voltage (volts)

$k_l$ = Deviation sensitivity (hertz/volt).

In binary FSK, the amplitude of the input signal can only be one of the two values, one for a logic 1 condition and other for a logic 0 condition. So, the peak frequency deviation is constant and always at its maximum value. Frequency deviation is simply plus or minus the peak voltage of the binary signal times the deviation sensitivity of the VCO. Because the peak voltage is the same for a logic 1 as it is for a logic 0, the magnitude of the frequency deviation is also the same for a logic 1 as it is for a logic 0.

**BFSK Receiver** – A simple circuit of FSK demodulator is shown in fig. 5.30. The FSK input signal is simultaneously applied to the inputs of both band-pass filters (BPFs) through a power splitter. The respective filter passes only the mark or only the space frequency on to its respective envelope detector. The envelope detectors, in turn, indicate the total power in each passband, and the comparator responds to the largest of the two powers. This type of FSK detection is called ***noncoherent detection*** because there is no frequency involved in the demodulation process that is synchronized either in phase, frequency or both with the incoming FSK signal.

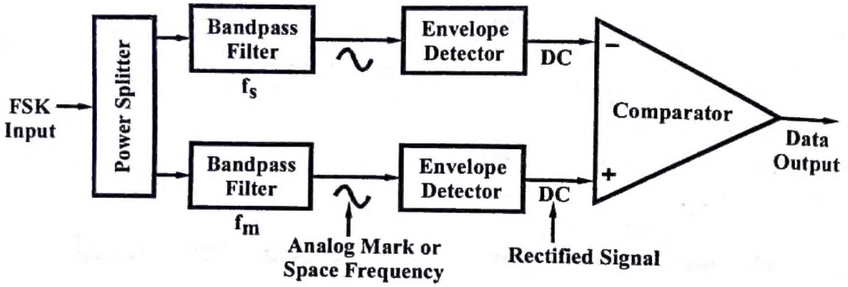


*Fig. 5.30 Noncoherent BFSK Demodulator*

The block diagram for a coherent BFSK receiver is shown in fig. 5.31. The incoming BFSK signal is multiplied by a recovered carrier signal which has the exact same frequency and phase as the transmitter reference. Generally, the two transmitted frequencies (the mark and space frequencies) are not continuous; it is not practical to regenerate a local reference which is coherent with both of them, due to which coherent BFSK detection is seldom used.

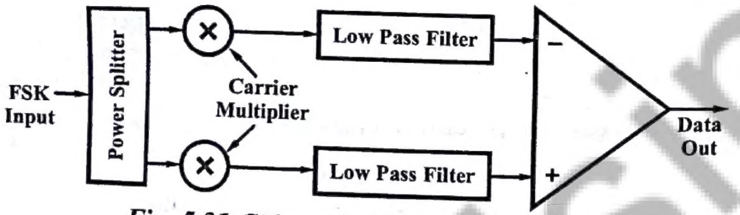


*Fig. 5.31 Coherent BFSK Demodulator*

Phase locked loop (PLL) is the most common circuit used for demodulating binary FSK signals, which is shown in block diagram form in fig. 5.32. A PLL-FSK demodulator works in a similar fashion as a PLL-FM demodulator. As the input to the PLL shifts between the mark and space frequencies, the D.C. error voltage at the output of the phase comparator follows the frequency shift. Since there are only two input frequencies (mark and space), there are also only two output error voltages. One represents a logic 1 and the other a logic 0. So, the output is a two-level (binary) representation of the FSK input. Usually, the natural frequency of the PLL is made equal to the center frequency of the FSK modulator. Consequently, the changes in the D.C. error voltage follow the changes in the analog input frequency and are symmetrical around 0V.

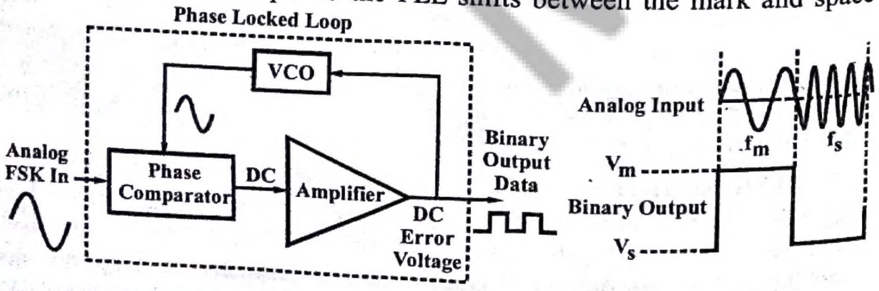


*Fig. 5.32 PLL-FSK Demodulator*

***Q.25. Write short note on BFSK modulation.*** ***(R.G.P.V., Nov. 2018)***

***Ans.*** Refer the ans. of Q.24.

***Q.26. State and prove Shannon's theorem for channel capacity.***

***Or***

***Explain the information capacity theorem for channel coding.***

***Or***

***Write short note on Shannon's theorem for channel capacity.***

***(R.G.P.V., Nov. 2018)***

***Ans.*** The information capacity of a continuous channel of bandwidth B Hz, disturbed by additive white Gaussian noise of power spectral density $N_0/2$ and limited in bandwidth to B, is given by

$$C = B \log\left(1+\frac{P}{N_0 B}\right) \text{ bits/sec}$$

where, P is the average transmitted power.

**Proof** – Consider a zero-mean, stationary process X(t), i.e., band-limited to B Hertz. Let $X_k$, k = 1, 2, ..., K, denote the continuous random variables obtained by uniform sampling of the process X(t) at the Nyquist rate of 2B samples per second. These symbol are transmitted in T seconds over a noisy channel, which is also bandlimited to B Hertz. Hence the number of samples, K, is given by

$$K = 2BT$$

$X_k$ refer as a sample of the transmitted signal. The channel output is corrupted by ***Additive White Gaussian Noise*** (AWGN) of zero mean and power spectral density $N_0/2$. The noise of the channel is band-limited to B Hz. Let the continuous random variables $Y_k$, k = 1, 2, ..., K indicate samples of the received signal as shown by

$$Y_k = X_k + N_k \text{ where, } k = 1, 2, \ldots, K \qquad \ldots(i)$$

where, $N_k$ is the noise sample with zero mean and variance given by $\sigma^2 = N_0B$ and the samples $Y_k$, k = 1, 2, ..., K are statistically independent.

Typically, the transmitter is power-limited, it is therefore reasonable to define the cost as

$$E[X^2_k] = P, \quad k = 1, 2, 3, \ldots, K \qquad \ldots(ii)$$

The information capacity of the channel is defined as the maximum of the mutual information between the channel input $X_k$ and the channel output $Y_k$ over all distribution on the input $X_k$ that satisfy the power constraints of equation (ii).

Let $I(X_k; Y_k)$ denote the average mutual information between $X_k$ and $Y_k$. Thus, the information capacity of the channel (or channel capacity) is as follows –

$$C = \max_{f_{X_k}(x)} \{I(X_k; Y_k) : E[X_k^2] = P\} \qquad \ldots(iii)$$

where, $f_{X_k}(x)$ is the probability density function of $X_k$.

For the evaluation of the information capacity C, we proceed in three stages –

(i) The variance of sample $Y_k$ of the received signal equals $P + \sigma^2$. Hence using the expression,

$$h(Y_k) = \frac{1}{2}\log_2(2\pi e\sigma^2) \qquad \ldots(iv)$$

We have the differential entropy of $Y_k$ as

$$h(Y_k) = \frac{1}{2}\log_2\left[(2\pi e(P + \sigma^2)\right] \qquad \ldots(v)$$

(ii) The variance of the noise sample $N_k$ equals $\sigma^2$. Hence, the use of equation (iv) gives the differential entropy of $N_k$ as

$$h(N_k) = \frac{1}{2}\log_2(2\pi e\sigma^2) \qquad \ldots(vi)$$

(iii) The average mutual information can be expressed as

$$I(X_k ; Y_k) = h(Y_k) - h(Y_k/X_k) \qquad \ldots(vii)$$

The maximization specified in equation (iii) is attained by choosing the sample of transmitted signal from a noiselike process of average power P. Thus, we reformulate equation (iii) as

$$C = I(X_k; Y_k) : X_k \text{ Gaussian}, \; E\left[X_k^2\right] = P \qquad \ldots(viii)$$

Substituting the equations (v) and (vi) into equation (vii) and recognizing the definition of information given in equation (viii), we get the desired result.

$$C = \frac{1}{2}\log_2\left(1 + \frac{P}{\sigma^2}\right) \text{ bits per transmission} \qquad \ldots(ix)$$

The channel used K times for the transmission of K samples of the process X(t) in T seconds. The information capacity per unit time is (K/T) times the result given in equation (ix).

Thus, we may express the information capacity per transmission as

$$C = B\log_2\left(1 + \frac{P}{N_0 B}\right) \text{bits per second.} \qquad \ldots(x)$$

The above equation (x) is for the capacity of the bandlimited channels, AWGN waveform channel with a band-limited average power limited input was first derived by Shannon in early 1948. It is also known as ***Shannon's third theorem.***

***Q.27. Discuss the Shannon limit.***

***Ans.*** Consider a Gaussian channel that is limited in bandwidth and power. We want to explore the communication system limits under these constraints. Let us define an ideal system that transmits data at a bit rate $R_b$ which is equal to the information capacity C. Then the average transmitted power is given as –

$$P = E_b C = E_b R_b \qquad \ldots(i)$$

where $E_b$ is the transmitted energy per bit. Therefore, the channel capacity theorem for ideal system can be defined as

$$\frac{C}{B} = \log_2\left(1 + \frac{E_b}{N_0}\frac{C}{B}\right) \qquad \ldots(ii)$$

This equation can be re-written in the following form as –

$$\frac{E_b}{N_0} = \frac{2^{C/B} - 1}{C/B} \qquad \ldots(iii)$$

The plot of the bandwidth efficiency $R_b/B$ versus $E_b/N_0$ is called the bandwidth-efficiency diagram, and is shown in fig. 5.33. The ideal system is shown by the line $R_b = C$.

Based on fig. 5.33, we can make the following conclusions –

(i) The ratio $E_b/N_0$ approaches the limiting value for infinite bandwidth

$$\left.\frac{E_b}{N_0}\right|_{B\to\infty} = \ln 2 = 0.693 = -1.6 \text{ dB} \qquad \ldots(iv)$$

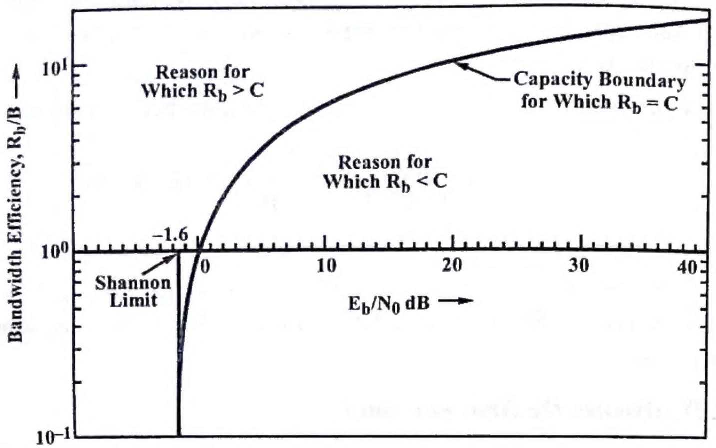


*Fig. 5.33 The Bandwidth Efficiency Diagram*

This value is called the Shannon limit. The channel capacity corresponding to this limiting value is expressed as –

$$C|_{B\to\infty} = \frac{P}{N_0}\log_2 e \qquad ...(v)$$

(ii) The curve for the critical rate $R_b$ = C is called the capacity boundary. Error-free communication is not possible for the case $R_b$ > C. However, for case $R_b$ < C, there exists some coding technique which can allow an arbitrarily low probability of error.

(iii) The diagram highlights potential trade-offs among $\frac{R_b}{B}, \frac{E_b}{N_0}$ and the probability of error $P_e$. Particularly, we may view operating point movement along a vertical line as trading $P_e$ versus $R_b/B$ for a fixed $E_b/N_0$. On the other hand, we may view operating point movement along a horizontal line as trading $P_e$ versus $E_b/N_0$ for a fixed $R_b/B$.

☯ ☯ ☯

RGPV

**B.E. (Third Semester) EXAMINATION June 2010**
**(New Scheme)**
**(Common for CS, EI & BM Engg. Branch)**
**DIGITAL CIRCUITS AND SYSTEMS**
**BM/CS/EI–303(N)**

**Note :** Attempt any *five* questions. All questions carry equal marks.

1. (a) Convert the following –

(i) $(B65F)_{16} = (\ )_{10}$ (ii) $(153.25)_{10} = (\ )_2$

(iii) $(10110001101011)_2 = (\ )_8$ (iv) $(153)_{10} = (\ )_8$.

**(See Unit-I, Page 17, Prob.7)**

(b) State and prove De-Morgan's theorem. **(See Unit-I, Page 46, Q.43)**

2. (a) Simplify the following Boolean function with K-map –

F (W, X, Y, Z) = Σ(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)

**(See Unit-I, Page 65, Prob.41)**

(b) Simplify the following using Quine and McCluskey's method –

F(W, X, Y, Z) = Σ(0, 1, 2, 8, 10, 11, 14, 15) **

3. (a) Design a full adder with the help of truth table. Explain the working of full adder by giving expressions for sum and carry in full adder.

**(See Unit-II, Page 78, Q.7)**

(b) Design a full subtractor using two half subtractor and an OR gate.

**(See Unit-II, Page 83, Q.13)**

4. (a) Design a BCD adder and also give the rules of BCD addition.

**(See Unit-II, Page 93, Q.29)**

(b) Explain the working of Look-ahead carry generator.

**(See Unit-II, Page 91, Q.26)**

5. (a) Explain the working of monostable multivibrators with the help of waveforms and circuit diagram. **(See Unit-IV, Page 211, Q.39)**

(b) Give a comparison of the following logic families –

DTL, RTL and TTL. **(See Unit-IV, Page 245, Q.76)**

6. (a) Design a 3 to 8 line decoder. **(See Unit-II, Page 105, Q.42)**

(b) Explain the working of multiplexer and draw the detailed circuit of 4 to 1 line multiplexer. **(See Unit-II, Page 98, Q.33)**

****Now, according to new revised syllabus of R.G.P.V., it is not included in syllabus**

**(b)** Discuss about the sample and hold circuits.

**(See Unit-IV, Page 203, Q.29)**

**8.** Write short notes (any four) –

(i) Shannon's theorem for channel capacity **(See Unit-V, Page 285, Q.26)**

(ii) BFSK modulation **(See Unit-V, Page 285, Q.25)**

(iii) Quantization error **(See Unit-V, Page 267, Q.15)**

(iv) Flash RAM **(See Unit-III, Page 179, Q.60)**

(v) Demultiplexer **(See Unit-II, Page 100, Q.36)**

(vi) Nyquist sampling theorem. **(See Unit-V, Page 251, Q.4)**

✤✤✤